DOI: https://doi.org/10.26642/ten-2024-1(93)-201-210
УДК 004.056

А.І. Оринчак, асистент
О.В. Кузьменко, ст. викладач
О.М. Свінцицька, к.е.н., доц.
Державний університет «Житомирська політехніка»

Виявлення загроз у реальному часі за допомогою JavaScript: механізми моніторингу та реагування

Розглянуто методи виявлення загроз у реальному часі, моніторинг та реагування на них, використовуючи JavaScript та безпекові механізми. Це дослідження описує можливості для покращення безпеки онлайн-додатків, застосовуючи можливості мови JavaScript. У статті розглядаються методи виявлення загроз у реальному часі та їх застосування в системах, що написано на JavaScript. Продемонстровано як безпечну розробку JavaScript і виявлення загроз у реальному часі, так і реагування на них. Показано, як описані методи можуть захищати вебдодатки у реальних випадках. Стаття має на меті дослідження JavaScript на предмет виявлення загроз у реальному часі, надає практичні рекомендації щодо моніторингу та легування за допомогою JavaScript, показує методи зменшення загроз у реальному часі.

Методологія починається зі всебічного вивчення літератури щодо виявлення загроз у реальному часі та безпеки онлайн-додатків. Приклади коду та дослідження випадків демонструють, як ці теми застосовуються на практиці. Це дослідження надає всебічне розуміння виявлення загроз у реальному часі за допомогою JavaScript, надаючи розробникам та фахівцям з безпеки інформацію для захисту вебдодатків від розповсюджених загроз.

Ключові слова: *виявлення загроз у реальному часі; безпека вебдодатків; моніторинг; механізми реагування; JavaScript.*

Актуальність теми. Онлайн-додатки стають невід'ємною частиною повсякденного життя в цифрову епоху, і потреба в надійних заходах безпеки важливіша, ніж будь-коли. З такою гнучкістю, що надається технологічним прогресом, важко повірити, що існують потенційні загрози для його розвитку.

Інтернет відкрив багато можливостей на критичному етапі цифрових технологій і продовжує бути незамінним у його використанні. Хоча цифрова трансформація та впровадження нових технологій сприяють подальшому розвитку, це також несе численні ризики. Розробники та кінцеві користувачі стикаються з серйозними перешкодами, які виникають через зростання цих ризиків.

З широким розповсюдженням інтернет-додатків важливо інтегрувати захист для безпечного переходу між провайдерами та кінцевими користувачами. Згідно з дослідженням 2021 року серед старших внутрішніх аудиторів у Великій Британії та Ірландії, існує розрив між внутрішньою аудиторською роботою, що надає гарантії щодо кібербезпеки, та оцінкою і просуванням культури кібербезпеки в їхніх організаціях.

Позитивний аспект результатів показав, що більшість старших внутрішніх аудиторів включають кібербезпеку в плани аудиту (81 %), ідентифікують загрози, перевіряють план пом'якшення (58 %) та проводять оцінку ризиків у співпраці з їхніми IT та ризик-колегами (62 %).

Однак у сферах, які безпосередньо сприяють впливовій культурі кібербезпеки, лише третина повідомляє про внесок у стратегію / політику кібербезпеки в організації (32 %), створення культури навчання на помилках (31 %) та оцінювання, чи організація інвестує в навчання з безпеки для працівників (33 %).

Опитування також підкреслило вплив пандемії коронавірусу на практики кібербезпеки та стійкість. Більше половини (51 %) респондентів повідомили, що зазнали кібератаки за останні 12 місяців, яка вплинула на продукти та послуги, і 42 % респондентів зазначили, що працівники, які працюють віддалено, створили бар'єр для впровадження кращих практик кібербезпеки. Однак, більш позитивно, 65 % старших внутрішніх аудиторів зазначили, що обговорення ризиків кібербезпеки відбувалися частіше з початку пандемії.

Виявлення загроз у реальному часі полегшує їх вирішення з такою швидкістю роботи, що захищає кібербезпеку від компроментації. JavaScript, популярна мова програмування, є доволі гнучкою у вирішенні завдань такого типу. Виявлення загроз у реальному часі, моніторинг та реагування стали необхідними для безпеки онлайн-додатків [1].

Сьогоднішні складні, динамічні та інтерактивні вебдодатки мають досить багато безпекових вразливостей, що робить це критичною проблемою [2].

Це дослідження має на меті запропонувати новий підхід до розслідування ролі, яку JavaScript відіграє в системах виявлення, моніторингу та реагування на загрози в реальному часі в онлайн-безпеці.

Безпекові механізми JavaScript містять процеси та інструменти, які використовуються для забезпечення безпеки JavaScript. Це включає виявлення цих вразливостей у додатках та вжиття заходів для їх усунення під час розробки або запобігання їх експлуатації в продукції.

Висновки дослідження також можуть надати важливу інформацію розробникам та організаціям, просвітивши їх щодо методів, стратегій та критичних практик, які представляють можливість захисту їх додатків від онлайн-загроз.

Дослідження також може надати глибоке розуміння того, як може бути досягнуто виявлення загроз у реальному часі, зміцнюючи безпеку та стійкість онлайн-додатків, використовуючи можливості JavaScript.

Також було зосереджено увагу на зміцненні захисту онлайн-додатків в епоху зростання кіберзагроз, роблячи внесок у постійне обговорення веббезпеки та пропонуючи ідеї щодо синергії між JavaScript та виявленням загроз у реальному часі.

Постановка проблеми. Розглянути методи виявлення кібербезпекових ризиків, описати функції для виявлення аномалій та аномалій на основі підписів на мові JavaScript,

Аналіз останніх досліджень та публікацій. *Історичний контекст та еволюція загроз.* Історія веб-безпеки багата прикладами того, як загрози та засоби захисту постійно змінювалися. Статичні HTML-сайти домінували на ранньому етапі розвитку інтернету, що становило відносно прості проблеми безпеки. Однак поверхня атаки значно збільшилася з впровадженням динамічних, керованих даними онлайн-додатків [1]. Поширення онлайн-технологій, контенту, створеного користувачами, та сторонніх інтерфейсів зумовило численні ризики. Загрози змінювалися з часом, переходячи від простих атак до більш складних експлойтів, таких як атаки розподіленої відмови в обслуговуванні (DDoS), SQL-ін'єкції та міжсайтові скрипти (XSS) [3].

Огляд літератури щодо виявлення загроз у реальному часі. Динамічний характер виявлення загроз у реальному часі показано завдяки ретельному аналізу сучасної літератури, що підкреслює важливість цієї галузі досліджень. Багато досліджень розглядають створення та використання методів на основі JavaScript для виявлення загроз та реагування на них. У [4] підкреслюється цінність систем логування та моніторингу в онлайн-безпеці та використання JavaScript для цих функцій. У дослідженні реальних аспектів виявлення загроз [7] акцентують увагу на необхідності швидких систем реагування при роботі зі швидкозмінними загрозами.

Як працює виявлення загроз у реальному часі. Перший і найпростіший метод полягає в моделюванні поведінки відомих кіберзагроз. Ці технології безпеки записують кожен файл, до якого шахраї намагалися отримати доступ. Також фіксується місце зберігання цих файлів. Крім того, в логах фіксується час спроби, а інші заходи безпеки можуть також записувати несанкціоновані спроби входу.

Інші системи можуть мати відомий список кіберзагроз або шкідливого програмного забезпечення. Технології кібербезпеки можуть легко ідентифікувати ці практики.

З 1990-х років ці автоматичні системи виявлення загроз стали нормою. Сьогоднішні загрози кібербезпеки значно складніші за ці порівняно прості методи.

Нижче наведено кілька широко використовуваних методів виявлення ризиків, що є поширеними сьогодні.

Аналітика поведінки користувачів та атак. Це більш складний спосіб спостереження за визнаними кіберзлочинами. Зберігання записів про відомі загрози кібербезпеки та дії є першим кроком. Це також створює надійну модель поведінки. Однак це надає точку відліку для системи виявлення загроз для ідентифікації аномалій. Користувачі, що знаходяться в Силіконовій долині, складатимуть більшість системи, призначеної для віддаленої роботи, головним чином використовуваної під час робочих годин за PST. Спроба входу з Сеулу буде повідомлена як підозріла діяльність і потребуватиме додаткового дослідження. У таких ситуаціях загрози безпеки можуть бути ідентифіковані та запобігти шляхом комбінування автоматичного виявлення ризиків з людськими аналітиками.

Створення пасток для зломщиків. Команди безпеки створюють обставини, які кіберзлочинці вважають надто спокусливими, щоб їх ігнорувати. Ми називаємо це пастками для зломщиків. Медова пастка (honeypot) – це один з типів пасток для зломщиків. Це може бути певний ресурс або актив, який вважається таким, що має мережеві ресурси або послуги. Хто намагається отримати доступ до цих ресурсів, робить це, виступаючи як загроза безпеці. Облікові дані для пасток honeypot – це інший приклад. За допомогою цих облікових даних ви можете отримати доступ до мережевих можливостей або вищих рівнів доступу. Команда безпеки знає про можливу загрозу безпеці, запросивши ці облікові дані. Якщо це вважатимуть за необхідне, вони можуть провести ручне розслідування.

Полювання на загрози. Полювання на загрози передбачає активний пошук ризиків безпеки, про які система ще не знає. Ваша мережа може бути систематично проаналізована за допомогою виявлення ризиків. Вона може оцінити кожен актив, ресурс, кінцеву точку, URL та обладнання на предмет наявності загроз безпеки. Це може включати трафік з незнайомих джерел, підозрілу мережеву

активність, наприклад, завантаження або зміну незвичайних файлів чи даних, спроби незаконного доступу або інші методи управління подіями.

Виявлення внутрішніх ризиків за допомогою штучного інтелекту (ШІ). Це безперервно відстежує кожного користувача і автоматично визначає незвичайну або ризикову активність за допомогою аналізу поведінки на основі ШІ. Це також визначає настрої, висловлені в текстах електронних листів, щоб визначити настрій автора. Для кожної особи під спостереженням створюється рейтинг ризику на основі різних сигналів та індикаторів, який відображається на одній панелі керування. Відстежуйте, як використовуються вебсайти та додатки, збирайте інформацію про активність і проактивно знаходьте будь-які фактори ризику. Автоматично фіксуйте з'єднання, встановлені додатками, а також використані порти та пропускну здатність. Спостерігайте за діями, пов'язаними з друком, знімними, хмарними та локальними сховищами. Спостерігайте за створенням, редагуванням, видаленням та перейменуванням файлів.

Методологія виявлення загроз у реальному часі. Використовує аналіз репутації URL та доменів для ідентифікації потенційно шкідливих посилань та вебсайтів. Ці системи порівнюють URL з відомими базами даних фішингу та блок-листами, оцінюючи їх репутацію та надійність. Система безпеки може легко виявити відомі загрози, і рішення для виявлення загроз в реальному часі можуть картографувати як відомі, так і невідомі загрози інфраструктури. Вони працюють, використовуючи розвідку загроз, встановлюючи пастки для вторгнення, досліджуючи підписні дані з попередніх атак та порівнюючи їх з реальними зусиллями вторгнення.

Експериментальна валідація. Сучасні підходи до архітектур безпеки реалізують цю ідею безпеки як процесу, забезпечуючи адекватні / пропорційні відповіді на різні загрози та підвищену видимість всієї системи та подій, що відбуваються в ній. Для моніторингу та аудиту рішень безпеки та підлягаючих фізичних і віртуальних систем використовуються спеціальні інструменти та програмні агенти. Перший підхід зосереджений на розумінні великої кількості зібраних даних логів за допомогою методів видобутку даних (DM), великих даних та AI / машинного навчання (ML). Другий шлях зосереджений на моделюванні шаблонів атак та спробах надати абстрактні описи для цих атак. Відповідно до цієї ідеї, Lockheed Martin Corporation розробила Cyber Kill Chain Framework у 2011 році – IT-переробку військового терміна «Find Fix Track Target Engage Assess» (F2T2EA). Ця теоретична основа має на меті покращити розуміння аналітиком тактик, технік та процедур противника, підвищуючи видимість нападу. Модель ідентифікує процеси розвідки, озброєння, доставки, експлуатації, встановлення, командування та управління та кібернападів. Автори також посилаються на доктрину операцій інформаційного департаменту оборони для зображення матриці дій за допомогою дій «виявляти, відмовляти, порушувати, деградувати, обманювати та знищувати», щоб визначити інструменти та засоби, які можуть бути використані для пом'якшення фаз кібернападу.

Потенційні виклики та майбутні напрямки. Сучасні підходи до архітектур безпеки реалізують цю ідею безпеки як процесу, забезпечуючи адекватні / пропорційні відповіді на різні загрози та підвищену видимість всієї системи та подій, що відбуваються в ній. Для моніторингу та аудиту рішень безпеки та підлягаючих фізичних і віртуальних систем використовуються спеціальні інструменти та програмні агенти. Інша проблема – зростання просунутих загроз, таких як кібератаки, які можуть використовувати вразливості в системах та інфраструктурі. Ці атаки стають більш складними та важко виявляються, що робить необхідним розгортання моделей кібербезпеки у реальному часі, які можуть швидко виявляти та запобігати їм.

Складність хмарних обчислень. Ми залежимо від хмари для багатьох наших повсякденних завдань і дій. Навіть до COVID-19 індустрія слідувала цим шляхом. Цей перехід був значно прискорений світовою пандемією та усіма логістичними труднощами, які з нею пов'язані. На сьогоднішній день хмарні обчислення використовуються не лише для зберігання файлів. Завдяки новим технологіям, таким як контейнери, додатки, а іноді й цілі системи можуть працювати в хмарі. Однак це створює безліч проблем для фахівців з кібербезпеки. Ця теоретична основа має на меті підвищити видимість нападу, допомагаючи аналітикам краще розуміти стратегії, техніки та процедури супротивника.

Зосередження на периметрі. Багато фахівців з кібербезпеки зосереджують свої зусилля на захисті периметра мережі. Це піддає організацію кільком проблемам безпеки. По-перше, багато сучасних кіберзагроз повністю обходять периметр. Фішинг та інші загрози безпеці часто проходять повз периметр. Необхідність посилення внутрішньої безпеки створює другий ризик. Після того як несанкціонована особа отримує доступ до вашої мережі, вона може отримати доступ майже до всього. Надмірне зосередження на периметрі мережі може також призвести до того, що деякі організації будуть вважати себе захищеними, коли це не так. Вони можуть припинити звертати увагу на безпеку своєї мережі, вважаючи, що їхня система безпечна.

Повільний час реагування. Хакери починають діяти негайно після випуску нового продукту або оновленої версії, намагаючись знайти та використати його слабкі місця. Незалежно від міцності рішення з кібербезпеки, воно має постійно адаптуватися до найновіших загроз кібербезпеки. Щоб уникнути будь-яких проблем, вам необхідно мати резервний план. Універсальні моделі типових поведінкових шаблонів

і атак можуть слугувати резервним периметром у разі, якщо ваша основна система виявлення ризиків не зможе ідентифікувати загрозу.

Відсутність інтегрованих інструментів. Багато інструментів кібербезпеки є вузькоспеціалізованими. Вони повинні бути розроблені таким чином, щоб працювати як єдиний комплекс. Це може призвести до різних проблем, які потребують вирішення, і викликати проблеми безпеки.

Зловмисне програмне забезпечення. Це найпоширеніший ризик кібербезпеки, від якого захищають інструменти виявлення загроз. Воно настільки поширене, що є стереотипним образом, який більшість людей уявляє, коли думають про виявлення загроз у реальному часі. Підозріле програмне забезпечення може бути завантажено з якогось джерела, можливо, з вебсайту, який ви відвідали. Після завантаження може з'явитися вікно із повідомленням про те, що завантаження було заблоковано. Ця ситуація також демонструє необхідність інструментів виявлення ризиків. Чи можете ви уявити серфінг в інтернеті без встановленого антивірусу? Зловмисне програмне забезпечення може бути різних форм. Однією з них є підозріле програмне забезпечення. Деякі занадто поширені типи зловмисного ПЗ – це віруси, троянські коні та шпигунські програми. Більшість рішень для виявлення загроз у реальному часі залишаються відносно актуальними щодо останніх зразків зловмисного програмного забезпечення, оскільки це очікувано. Як зазначалося раніше, кіберзлочинці завжди будуть швидшими за IT-фахівців. Інструменти безпеки для виявлення загроз також мають стежити за вторинними ознаками зловмисного програмного забезпечення. З огляду на темпи, з якими з'являються нові рішення з кібербезпеки у відповідь на ці ризики, зрозуміло, що в майбутньому з'явиться багато нових стратегій для захисту даних.

Прогалини та обмеження в сучасних дослідженнях. Хоча виявлення загроз у реальному часі за допомогою JavaScript значно просунулося, залишаються кілька прогалин і обмежень. По-перше, хоча було проведено багато досліджень щодо різних методів виявлення загроз, немає добре продуманої структури для впровадження цих методів у онлайн-додатки, керовані JavaScript. Це створює труднощі для розробників, які прагнуть реалізувати практичні ідеї [5]. Різноманітність та складність онлайн-додатків ускладнюють створення загальних рішень, які функціонують у багатьох сценаріях. Необхідний більш гнучкий і всебічний план. Нарешті, враховуючи динамічний характер середовища загроз, необхідно вдосконалити дослідження щодо поточної розвідки загроз та адаптації в системах виявлення загроз у реальному часі [6]. Через ці прогалини в літературі ця стаття є значущою. Вона розглядає ці обмеження, пропонуючи глибоке розуміння ролі JavaScript та практичних рішень для виявлення загроз у реальному часі та реагування на них у контексті змінюваних викликів веббезпеки.

Викладення основного матеріалу. *JavaScript* є гнучкою та повсюдною мовою програмування, яка є необхідною для сучасної веброзробки. Вона покращує взаємодію з користувачем та онлайн-безпеку. У цьому розділі розглядається подвійна функція JavaScript у веббезпеці – покращення безпеки та створення вразливостей. JavaScript є критичним компонентом веброзробки, оскільки дозволяє створювати адаптивні інтерфейси користувача, асинхронне отримання даних та динамічний і інтерактивний контент [8]. Завдяки виконанню на стороні клієнта JavaScript є важливою частиною сучасних вебдодатків. JavaScript забезпечує валідацію форм, автентифікацію користувачів та зміни в реальному часі, що покращує взаємодію з користувачем [10]. Зловмисники націлюються на JavaScript через його популярність та можливість використовувати його функції.

JavaScript як інструмент та вразливість безпеки. Подвійна природа JavaScript робить його надзвичайно важливим для онлайн-безпеки. Технологія JavaScript дозволяє розробникам включати функції безпеки, такі як системи виявлення загроз у реальному часі та системи реагування. Завдяки своїй адаптивності можуть бути розроблені функції безпеки на стороні клієнта, враховуючи безпечну автентифікацію, контроль доступу та валідацію введення даних [9]. JavaScript відіграє важливу роль у забезпеченні міцних заходів безпеки вебдодатків.

Однак JavaScript також може бути джерелом загроз безпеці. Атаки на основі міжсайтових скриптів (XSS) використовують JavaScript для виконання неперевірених введень користувачів у вигляді скриптів на вебсторінці, що може призвести до витоку даних користувачів або поширення шкідливого програмного забезпечення [10]. Динамічна природа JavaScript та велика кількість сторонніх модулів і залежностей створюють можливості для експлуатації.

Поточні найкращі практики для безпечної розробки JavaScript. Дотримання останніх рекомендацій щодо безпечної розробки JavaScript є важливими для зменшення ризиків, пов'язаних з цією мовою програмування. Серед цих методів:

- валідація введення. Для запобігання XSS-атакам ретельно перевіряйте введення користувачів [8]. Переконайтеся, що дані з ненадійних джерел очищені та перевірені перед використанням;
- реалізація заголовків політики безпеки контенту (CSP). Використовуйте CSP-заголовки для управління тим, які скрипти дозволено виконувати на вебсторінці [10]. Це зменшує шкоду, яку можуть завдати шкідливі програми;
- використання бібліотек. Виправляйте відомі вразливості безпеки та регулярно оновлюйте та патчуйте сторонні бібліотеки та залежності [9];

- строгий контроль доступу та процедури автентифікації. Реалізуйте строгі контролі доступу та процедури автентифікації, щоб гарантувати, що тільки ті, хто має дозвіл, можуть отримати доступ до певних ресурсів [9];

- дотримання безпечних практик кодування. Уникайте використання функції eval(), використовуйте строгий режим (strict mode) та зменшуйте використання глобальних змінних [10].

Розробники мають використовувати потенціал JavaScript як ефективного інструменту для захисту вебдодатків і зменшення їх вразливості до експлуатації, дотримуючись цих рекомендованих практик.

Техніки виявлення загроз у реальному часі. Веббезпека потребує виявлення загроз у реальному часі для ідентифікації та пом'якшення атак. У цьому розділі розглядаються методи виявлення загроз у реальному часі та їх використання в JavaScript. Ці методи демонструються за допомогою прикладів та фрагментів коду.

Виявлення аномалій. Виявлення аномалій передбачає моніторинг поведінки системи для виявлення відхилень від типового або очікуваного стану [11]. Цей метод особливо корисний для виявлення загроз у реальному часі, оскільки він може виявляти дивні шаблони або поведінки, які можуть свідчити про активну атаку. Наприклад, можна застосовувати методи машинного навчання для аналізу шаблонів взаємодії користувачів та виявлення аномалій у середовищі, де використовується JavaScript. Прикладами аномалій можуть бути незвичні звички входу користувачів, дивні шаблони доступу до даних або несподівані передачі даних.

Приклад виявлення аномалій за допомогою JavaScript:

```
function detectAnomalies(userActivity) {
  // Псевдокод для реалізації алгоритму машинного навчання
  // Аналізувати дані активності для виявлення аномалій
  let anomalyDetected = false;

  // Приклад: якщо активність користувача перевищує певний поріг
  if (userActivity.loginAttempts > 10 || userActivity.dataAccessPattern == 'unusual') {
    anomalyDetected = true;
  }

  if (anomalyDetected) {
    // Вжити відповідні заходи в реальному часі
    alert('Аномалія виявлена у діяльності користувача!');
    // Записати та повідомити про виявлену аномалію
    logAnomaly(userActivity);
    reportAnomaly(userActivity);
  }
}

function logAnomaly(activity) {
  console.log('Аномалія виявлена:', activity);
}

function reportAnomaly(activity) {
  // Відправити звіт про аномалію адміністраторам системи
  alert('Повідомлення адміністраторам про аномалію.');
}

// Приклад використання функції
let userActivity = {
  loginAttempts: 15,
  dataAccessPattern: 'unusual'
};

detectAnomalies(userActivity);
```

Виявлення аномалій є потужним інструментом для виявлення загроз у реальному часі. Використовуючи методи машинного навчання та аналізуючи поведінку користувачів, розробники можуть виявляти відхилення, які можуть свідчити про потенційні загрози, та вживати відповідні заходи для їх пом'якшення.

Виявлення на основі підписів. За словами Скарфоне [16], виявлення на основі підписів порівнює відомі шаблони атак, або підписи, з вхідними даними або запитами. Ця технологія може використовуватися в реальному часі шляхом порівняння вхідних даних з базою даних відомих підписів атак. Вхідні дані можуть бути оброблені за допомогою JavaScript, а методи порівняння підписів можуть бути використані.

Приклад виявлення на основі підписів за допомогою JavaScript:

```
function detectSignatures(requestData) {
  const knownSignatures = getKnownSignatures(); // Отримати відомі підписи атак

  for (const signature of knownSignatures) {
    if (requestData.includes(signature)) {
      // Вжити заходів у реальному часі, коли виявлено підпис
      alert('Підпис атаки виявлено: ' + signature);
      // Записати та заблокувати запит, наприклад
      logAttack(signature, requestData);
      blockRequest();
      break; // Після виявлення підпису припинити подальший аналіз
    }
  }
}

function getKnownSignatures() {
  // Повернути масив відомих підписів атак
  return ['signature1', 'signature2', 'signature3'];
}

function logAttack(signature, requestData) {
  console.log('Атака виявлена:', signature, requestData);
}

function blockRequest() {
  alert('Запит заблоковано.');
}

// Приклад використання функції
let requestData = "example requestData containing signature1";
detectSignatures(requestData);
```

Виявлення на основі підписів є ефективним методом для виявлення відомих атак у реальному часі. Використовуючи JavaScript для порівняння вхідних даних з базою даних відомих підписів, розробники можуть швидко ідентифікувати загрози та вживати відповідні заходи для їх блокування та запису.

Моніторинг та логування в JavaScript. Значення логування та моніторингу в контексті виявлення загроз неможливо переоцінити. Ці процедури необхідні для збору даних, виявлення аномалій та виявлення можливих порушень безпеки в реальному часі. У цьому розділі розглядається важливість моніторингу та логування у виявленні загроз, а також представлені бібліотеки JavaScript та інструменти для ефективного моніторингу й логування в реальному часі. Також надані приклади коду для налаштування цих функцій у додатках на JavaScript.

Інтенсивні процедури моніторингу та логування є необхідними для ефективного виявлення загроз у реальному часі. Потенційні ризики можуть бути виявлені на ранніх стадіях шляхом моніторингу різних аспектів вебдодатків, враховуючи взаємодії користувачів, запити серверів та продуктивність системи [15]. Крім того, повні логи є безцінним інструментом для підтримки судово-медичних досліджень, відстеження джерела атак та аналізу інцидентів безпеки [14].

Бібліотеки та інструменти JavaScript для моніторингу й логування в реальному часі. Моніторинг та логування в реальному часі в онлайн-додатках можуть бути реалізовані за допомогою кількох фреймворків та інструментів JavaScript, враховуючи:

1. Winston. Winston є популярним пакетом логування для JavaScript, який працює з кількома транспортами, такими як файли, консолі та віддалені сервери. Оскільки він дозволяє розробникам змінювати рівні та формати логів, його можна використовувати для логування в реальному часі у вебдодатках [21];

2. ELK Stack (Elasticsearch, Logstash, Kibana). Elasticsearch, Logstash та Kibana (ELK) є потужним рішенням з відкритим кодом для аналізу логів у реальному часі та їх централізованого ведення. Розробники можуть збирати логи, аналізувати їх та створювати візуалізації для моніторингу та виявлення загроз, інтегруючи додатки JavaScript з Elastic Logs [22];

3. Prometheus ma Grafana. Prometheus є широко використовуваним інструментом моніторингу та оповіщення, тоді як Grafana є інструментом візуалізації з відкритим кодом. Ці інструменти можуть використовуватися для моніторингу продуктивності та використання ресурсів додатків JavaScript у реальному часі. Ви можете виявляти аномалії та вживати заходів у відповідь на можливі загрози, налаштовуючи індивідуальні оповіщення та панелі управління [23, 24].

Механізми реагування. Здатність швидко та точно реагувати на виявлені загрози є критично важливою в галузі онлайн-безпеки. Для повного виявлення загроз у реальному часі необхідний встановлений план реагування. У цьому розділі розглядаються кілька методів реагування, які можна використовувати у разі виявлення небезпеки, обговорюється реалізація логіки реагування за допомогою JavaScript і підкреслюється важливість швидкої та точної реакції для пом'якшення ризиків у реальному часі.

Плани реагування мають бути впроваджені для мінімізації можливих пошкоджень та захисту цілісності онлайн-додатків, коли загрози виявляються в реальному часі. Кілька типових механізмів реагування:

1. Блокування та карантин. Негайне блокування або карантин можуть бути застосовані у відповідь на виявлені ризики, такі як підозріла поведінка користувача або шкідливі запити. Це може зупинити подальшу взаємодію користувачів з програмою або ізолювати загрозу для подальшого аналізу [16];

2. Попередження та сповіщення. Адміністратори та команди безпеки можуть отримувати сповіщення та попередження в реальному часі, що надає їм ранні повідомлення про потенційні загрози. Згідно з [3] ці сповіщення можуть ініціювати розслідування або попередньо визначені методи реагування на інциденти;

3. Перевірка автентичності користувачів. У разі підозрілої або аномальної активності користувачів можуть бути запропоновані виклики, такі як двофакторна автентифікація або CAPTCHA, для підтвердження особи та намірів користувача [14];

4. Припинення сесії. Підозрілі сесії можуть бути завершені, щоб запобігти подальшому незаконному доступу та захистити облікові записи користувачів та дані [1].

Реалізація логіки реагування за допомогою JavaScript. JavaScript є корисним інструментом для реалізації логіки реагування в реальному часі. Коли загрози виявляються, вони можуть використовуватися для вжиття відповідних заходів.

Приклад логіки реагування за допомогою JavaScript:

```
function handleThreatDetected(threatType) {
  if (threatType === 'XSS') {
    // Блокувати шкідливий запит та повідомити адміністраторів
    blockRequest();
    sendAlertToAdmins('Виявлено атаку XSS.');
  } else if (threatType === 'SuspiciousActivity') {
    // Вимагати від користувача пройти CAPTCHA
    promptUserForCAPTCHA();
  } else {
    // Записати загрозу та вжити відповідних заходів
    logThreat(threatType);
  }
}

function logThreat(threatType) {
  console.log('Виявлено загрозу:', threatType);
}

// Приклад використання функції
handleThreatDetected('XSS');
```

У цьому прикладі тип загрози визначається за допомогою JavaScript, після чого здійснюється відповідний механізм реагування. Це може включати блокування загрози, виклик для користувача або запис загрози для подальшого аналізу.

Важливість швидких та точних відповідей. Швидка та точна реакція на виявлені загрози є важливою з кількох причин:

– *мінімізація шкоди*. Швидкі дії можуть зменшити можливу шкоду, яку може завдати загроза, включаючи незаконний доступ або перехоплення конфіденційної інформації [10];

– *запобігання ескалації*. Швидка реакція може запобігти перетворенню загроз на більш серйозні інциденти безпеки, які можуть бути складнішими та дорожчими для вирішення [2];

– *збереження довіри користувачів*. Швидкі та точні відповіді допомагають зберегти довіру користувачів, демонструючи відданість безпеці та захисту даних [16];

– *мінімізація простоїв*. Онлайн-додатки можуть уникнути простоїв та зберегти доступність для користувачів, швидко реагуючи на загрози [15].

Швидка та точна реакція на виявлені загрози є критично важливою для забезпечення безпеки онлайн-додатків. Вона дозволяє мінімізувати можливу шкоду, запобігти ескалації загроз, зберегти довіру користувачів та мінімізувати простої. Реалізація ефективних механізмів реагування за допомогою JavaScript допомагає розробникам забезпечити швидке та точне реагування на загрози.

Дослідження випадків та приклади. Два реальних приклади покажуть практичність технологій виявлення загроз та реагування у реальному часі на основі JavaScript.

Дослідження випадку 1: Пом'якшення атак XSS.

Сценарій. Атаки XSS на відомий сайт електронної комерції почастішали, змінюючи відгуки користувачів для обману покупців та псування сторінок продуктів.

Механізм реагування. Команда безпеки використовувала JavaScript для виявлення та реагування в реальному часі. Після виявлення підозрілого скриптового коду у введенні користувача програма зупиняла запит, повідомляла про інцидент та сповіщала адміністраторів.

Результати. Виявлення та реагування у реальному часі зменшили кількість атак XSS. Захист відгуків користувачів та сторінок продуктів від вразливостей був можливий завдяки швидкому фільтруванню шкідливих введень.

Дослідження випадку 2: Запобігання атакам методом брутфорс.

Сценарій. У банківській інфраструктурі фінансової установи почастішали спроби брутфорс-атак на входи, що компрометували безпеку облікових записів.

Механізм реагування. Було впроваджено моніторинг входів у реальному часі за допомогою JavaScript. Якщо система виявляла кілька невдалих спроб входу з однієї IP-адреси протягом короткого періоду, обліковий запис тимчасово блокувався.

Результати. Блокування в реальному часі значно зменшило успіх брутфорс-атак. Неможливість повторних спроб входу захистила облікові записи користувачів від незаконного доступу.

Аналіз ефективності. Дослідження випадків показують, що технології виявлення загроз у реальному часі та реагування на основі JavaScript працюють. Інтегруючи JavaScript у свої плани безпеки, підприємства можуть швидко та точно реагувати на нові загрози, мінімізуючи шкоду та захищаючи цілісність онлайн-додатків. Переваги цих досліджень випадків:

- зменшення кількості успішних атак. Технології виявлення загроз у реальному часі та реагування на основі JavaScript зменшили кількість успішних атак, захищаючи цілісність даних та довіру користувачів;

- швидке пом'якшення загроз. Швидка реакція на загрози запобігає подальшій експлуатації та ескалації, захищаючи додаток;

- безпечність для користувачів. Двофакторна автентифікація, тимчасове блокування та інші зручні для користувачів заходи безпеки допомогли підприємствам покращити безпеку без незручностей для законних користувачів;

- доступність даних. Ці методи зберегли доступність онлайн-послуг, зменшуючи ризик атак.

Виклики та перспективи подальших досліджень. Системи виявлення загроз та реагування в реальному часі на основі JavaScript працюють, але мають недоліки. У цьому розділі розглядаються ці питання та пропонуються дослідження в галузі онлайн-безпеки та їх вирішення.

Виклики та обмеження:

- швидка еволюція ландшафту загроз. Це постійна проблема. Виявлення загроз в реальному часі має адаптуватися до нових методів атак та вразливостей [17];

- хибні спрацьовування. Виявлення загроз у реальному часі може помилково ідентифікувати законну активність користувачів. Управління цією проблемою без зниження рівня безпеки є складним завданням [18];

- використання ресурсів. Моніторинг та реагування в реальному часі потребують значної кількості системних ресурсів. Потрібно ефективно зменшити накладні витрати ресурсів [19];

- безпека крос-доменних сценаріїв. Забезпечення безпеки крос-доменних сценаріїв на основі JavaScript є складним завданням. У цих обставинах важко забезпечити правильне та безпечне виконання механізмів реагування [8].

Майбутні дослідження та вдосконалення:

- машинне навчання та ШІ. Майбутні дослідження можуть покращити виявлення загроз у реальному часі, інтегруючи машинне навчання та ШІ. Ці системи можуть навчатися нових загроз, підвищуючи точність та знижуючи кількість хибних спрацьовувань [11];

- інтеграція розвідки загроз. Інтеграція розвідувальних даних про загрози з виявленням загроз у реальному часі може надати актуальну інформацію про загрози. Це може покращити виявлення та реагування на загрози [7];

- обчислення на краю мережі. Виявлення загроз у реальному часі за допомогою обчислень на краю мережі може зменшити використання ресурсів та підвищити швидкість реагування. Пристрої на краю мережі можуть попередньо обробляти дані перед їх надсиланням на сервер [15];

- безперервний моніторинг. Моніторинг більшої кількості компонентів вебдодатків, враховуючи сторонні бібліотеки та API, може допомогти виявити та пом'якшити зовнішні загрози [14];

- стандартизація та найкращі практики. Створення галузевих стандартів та рекомендацій щодо найкращих практик для виявлення загроз та реагування на основі JavaScript може допомогти розробникам застосовувати їх [10].

Ці питання та дослідницькі ініціативи можуть сприяти розвитку онлайн-безпеки, покращуючи виявлення загроз у реальному часі та реагування у середовищах, керованих JavaScript.

Висновки. У статті було розглянуто виявлення загроз у реальному часі за допомогою JavaScript у вебдодатках. Наголошено важливість описаного підходу у швидко змінюваному світі цифрової безпеки. Розглянуто історичний контекст подвійної ролі JavaScript та найкращі практики для безпечної розробки. Виявлення загроз у реальному часі було представлено за допомогою прикладів коду. Було визначено значення моніторингу, логування та механізмів реагування, підкреслюючи важливість швидких та точних дій. Дослідження випадків показали, як ці методи захищають онлайн вебдодатки та довіру користувачів. Було визначено виклики та обмеження, а також запропоновано вдосконалення. Ця стаття робить висновок, що виявлення загроз у реальному часі за допомогою JavaScript є необхідним для безпеки онлайн-додатків. Згідно з результатами, швидкі та точні системи реагування зменшують шкоду та зберігають доступність онлайн-сервісів. В подальшому планується використовувати мову JavaScript для підвищення онлайн-безпеки та збільшення довіри користувачів.

References:

1. Smith, A. (2019), «Cybersecurity in the Digital Age: Challenges and Solutions», *Journal of Cybersecurity*, Vol. 5, No. 3, pp. 112–127.
2. Jones, M.A. and Johnson, P.R. (2020), «Web Application Security: Trends and Challenges», *Security Trends*, Vol. 17, No. 1, pp. 23–38.
3. Brown, A. and Johnson, S. (2021), «Emerging Threats in Web Security: A Comprehensive Analysis», *Journal of Cybersecurity Research*, Vol. 8, No. 2, pp. 45–61.
4. Williams, L. (2018), «Monitoring and Logging in Web Applications: A Comprehensive Guide», *Web Security Journal*, Vol. 12, No. 1, pp. 33–48.
5. Martin, R. and Lee, C. (2022), «A Granular Approach to Real-Time Threat Detection in Web Applications», *Proceedings of the International Conference on Web Security (ICWS)*, pp. 167–180.
6. Robinson, S. (2020), «Enhancing Web Application Security: Current Challenges and Future Directions», *Security Trends*, Vol. 22, No. 4, pp. 113–128.
7. Davis, P. and Smith, J. (2018), «Real-Time Threat Detection in Dynamic Web Environments», *Web Security Quarterly*, Vol. 15, No. 3, pp. 87–101.
8. Flanagan, D. (2020), *JavaScript: The Definitive Guide*, 7th ed., O'Reilly Media, 706 p.
9. Howard, M. (2002), *Writing Secure Code*, Microsoft Press, 477 p.
10. Resig, J., Bibeault, B. and Maras, J. (2016), *Secrets of the JavaScript Ninja*, Manning Publications, 464 p.
11. OWASP (2020), «OWASP Top Ten Project», [Online], available at: https://owasp.org/www-project-top-ten/
12. Moustafa, N. and Slay, J. (2015), «The ADFA Intrusion Detection Datasets», *Proceedings of the Australasian Computer Science Week Multiconference (ACSW)*, pp. 1–10.
13. Scarfone, K. and Mell, P. (2007), *Guide to Intrusion Detection and Prevention Systems (IDPS)*, National Institute of Standards and Technology (NIST), pp. 1–127.
14. Mena, J. et al. (2015), «A Survey of Big Data Architectures and Machine Learning Algorithms in Large-Scale Data Predictive Analytics», *Journal of King Saud University – Computer and Information Sciences*.
15. Liu, S., Liu, C. and Yao, D. (2018), «Web Application Security: Threats, Countermeasures, and Beyond», *Journal of Cybersecurity Research*, Vol. 10, No. 1, pp. 45–63.
16. Dempsey, K. et al. (2011), *Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations*, NIST Special Publication 800-137, National Institute of Standards and Technology, pp. 1–80.
17. Scarfone, K. and Mell, P. (2019), *Guide to Intrusion Detection and Prevention Systems (IDPS)*, NIST Special Publication 800-94, National Institute of Standards and Technology (NIST).
18. Brown, A. and Smith, E. (2022), «Practical Approaches to Real-time Threat Detection with JavaScript», *Proceedings of the International Conference on Web Security (ICWS)*, pp. 167–180.
19. Johnson, M. and Adams, R. (2019), «Web Application Security: Current Challenges and Future Directions», *Security Trends*, Vol. 26, No. 1, pp. 33–48.
20. Robinson, S. (2020), «Enhancing Web Application Security: Current Challenges and Future Directions», *Security Trends*, Vol. 22, No. 4, pp. 113–128.

21. Straub, J. (2020), «Modeling Attack, Defense and Threat Trees and the Cyber Kill Chain, ATT&CK and stride Frameworks as Blackboard Architecture Networks», *IEEE International Conference on Smart Cloud (SmartCloud)*, pp. 148–153.
22. Winston, [Online], available at: https://github.com/winstonjs/winston/
23. Elastic, [Online], available at: https://www.elastic.co/guide/index.html
24. Prometheus, [Online], available at: https://prometheus.io/docs/introduction/overview/
25. Graphana, [Online], available at: https://grafana.com/docs/

Оринчак Андрій Іванович – асистент кафедри комп'ютерних наук Державного університету «Житомирська політехніка».

https://orcid.org/0009-0004-5993-7129

Наукові інтереси:

– вебтехнології;
– кібербезпека.

Кузьменко Олександр Вікторович – старший викладач Державного університету «Житомирська політехніка».

https://orcid.org/0000-0002-4937-3284.

Наукові інтереси:

– інтелектуальний аналіз даних;
– веб-розробка.

Свінцицька Олександра Миколаївна – кандидат економічних наук, доцент, доцент кафедри комп'ютерних наук Державного університету «Житомирська політехніка».

https://orcid.org/0000-0002-2613-2437

Наукові інтереси:

– управління ІТ-проєктами;
– інформаційні системи і технології в креативних індустріях.

Orynchak A.I., Kuzmenko O.V., Svintsytska O.M.

Real-Time Threat Detection with JavaScript: Monitoring and Response Mechanisms

Web apps are essential to contemporary life but are also vulnerable to many security concerns. This article examines real-time threat detection, monitoring, and response techniques at the confluence of JavaScript and security. This study improves online application security by using JavaScript's unique characteristics. This paper examines real-time threat detection methods and their application in JavaScript-driven systems. We demonstrate safe JavaScript development and real-time threat detection and response. We show how these methods protect web applications in real-world cases. This article aims to explore JavaScript for real-time threat detection, give practical advice on JavaScript monitoring and logging, display real-time threat mitigation techniques. The methodology begins with a comprehensive literature study on real-time threat detection and online application security. Code examples and case studies demonstrate how the topics are used. This research provides a comprehensive understanding of real-time threat detection with JavaScript, equipping developers and security practitioners to protect web applications from evolving threats.

Keywords: real-time threat detection; web security, monitoring, response mechanisms, JavaScript.

Стаття надійшла до редакції 20.05.2024.