Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет комп'ютерних наук
Спеціальність 125 «Кібербезпека»
Освітня програма «Безпека інформаційних та комунікаційних систем»

«Допущено до захисту»
Зав.кафедрою БІСТ
Сергій РАССОМАХІН

_____
« » 2022 р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Аналіз ефективних методів контролю якості та пошук вразливостей у сучасних клієнт-серверних веб-додатках»

оцінка « » Керівник к.т.н. Мелкозьорова О.М.
Голова ЕК Рецензент проф., д.т.н. Краснобаєв В.А.
Доценко С.І. _____ Виконавець : студент групи КБ-61
Яремчук К. С.

Харків – 2022

РЕФЕРАТ

Звіт про виконання дипломної роботи: 65 сторінок, 16 рисунків, 2 формули, 3 таблиці, 1 додаток та 30 джерел.

Мета роботи полягає у аналізі ефективних методів контролю якості та пошуку вразливостей у сучасних клієнт-серверних веб-додатках, а також виробленні практичних висновків на основі отриманих результатів й наданні рекомендацій щодо вибору певного інструменту виявлення вразливостей.

У роботі було розглянуто та проаналізовано 5 сучасних інструментів контролю якості веб-додатків з можливістю сканування систем на недоліки з боку безпеки. Проведено порівняльний аналіз результатів для виявлення найефективніших програмних рішень, а також розглянута ефективність їх практичного використання. На підставі проведеного порівняльного аналізу вироблено практичні висновки щодо доцільності використання розглянутих сканерів на практиці, а також надано рекомендації щодо вибору кращого інструменту для отримання найбільш точних результатів сканування веб-додатку.

Результати виконання цього дослідження можуть бути використані як загальний огляд інструментів забезпечення безпеки веб-додатків, задля ознайомлення з існуючими сучасними методами контролю якості та пошуку вразливостей, а також в якості практичних рекомендацій щодо вибору того чи іншого сканеру для тестування веб-систем на предмет проблем безпеки.

Можливими напрямками розвитку є огляд платних інструментів контролю якості й пошуку вразливостей веб-додатків, в яких ефективність та точність виявлення вразливостей може бути порівняно кращою за безкоштовні інструменти.

Ключові слова: МЕТОДИ КОНТРОЛЮ ЯКОСТІ, КЛІЄНТ-СЕРВЕРНІ ВЕБ-ДОДАТКИ, СКАНУВАННЯ ВЕБ-ДОДАТКІВ, ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ, OWASP TOP TEN, СКАНЕРИ ВЕБ-ДОДАТКІВ.

ABSTRACT

Thesis report: 65 pages, 16 figures, 2 formulas, 3 tables, 1 appendix and 30 sources.

The purpose of the paper is to analyze effective methods of quality control and finding vulnerabilities in modern client-server web applications, as well as to develop practical recommendations for choosing a specific vulnerability detection tool.

The paper considered and analyzed 5 modern quality control tools for web applications with the ability to scan systems for security flaws. A comparative analysis of the results was conducted to identify the most appropriate software solutions, as well as the effectiveness of their practical use was considered. On the basis of the comparative analysis, conclusions were made about the feasibility of using the considered scanners in practice in modern conditions, as well as practical recommendations were made for choosing the best tool for obtaining the most accurate results of scanning a web application.

The results of this study can be used as a general overview of web application security tools, for familiarization with existing modern methods of quality control and vulnerability detection, as well as as practical recommendations for choosing a particular scanner for testing web systems for security problems.

Possible areas of development include the review of paid tools for quality control and vulnerability detection of web applications, in which the effectiveness and accuracy of vulnerability detection may be comparatively better than free tools.

Keywords: QUALITY CONTROL METHODS, CLIENT-SERVER WEB APPLICATIONS, WEB APPLICATION SCANNING, VULNERABILITY DETECTION, OWASP TOP TEN, WEB APPLICATION SCANNERS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AJAX – Asynchronous Javascript and XML
API – Application Programming Interface
CI/CD – Continuous Integration/Continuous Delivery
CSRF – Cross-site request forgery
CSS – Cascading Style Sheets
CVE – Common Vulnerabilities and Exposures
DAST – Dynamic Application Security Testing
DOM – Document Object Model
DVWA – Damn Vulnerable Web Application
HTML – HyperText Markup Language
HTTP – HyperText Transfer Protocol
HTTPS – HyperText Transfer Protocol Secure
IP – Internet Protocol
JSON – JavaScript Object Notation
OWASP – Open Web Application Security Project
RDP – Remote Desktop Protocol
REST – Representational State Transfer
SAST – Static Application Security Testing
SQL – Structured Query Language
SSH – Secure Shell
SSL – Secure Sockets Layer
SSRF – Server-side request forgery
TCP – Transmission Control Protocol
URL – Uniform Resource Locator
XML – Extensible Markup Language
XSS – Cross-Site Scripting

YAML – YAML Ain't Markup Language

ZAP – Zed Attack Proxy

БД – База даних

ПЗ – Програмне забезпечення

ВСТУП

Завдяки тривалому часу свого розвитку в умовах швидкорозвиваючихся мережевих систем й ринку інтернет комерції, веб-додатки почали являти собою дещо куди значніше, аніж просто сайти з контентом. На просторах мережі Інтернет с кожним днем з’являються все більш складні веб-додатки за своїми цілями й можливостями, котрі пропонують нові рішення задля задоволення вимог споживачів в усіх галузях мережі. Веб-додатки являють собою найбільш зручний та ефективний засіб для представлення інформації й надання послуг у мережі Інтернет. Компанії з різнобічних галузей ринку продовжують створювати веб-додатки для просування своїх товарів та послуг у Інтернеті, займаючи свої ніші у цифровому світі.

Мобільні інструменти та веб-технології захопили вершину списку на довгі роки. Такі цифрові сервіси часто бувають критично значущими й потребують відповідного захисту, задля забезпечення безпеки різного роду конфіденційної інформації та даних як користувачів зокрема, так і систем в цілому. Галузь веб-технологій розвивається невпинними кроками і тому несе за собою певні ризики. Гарантій безпеки та захищеності системи з такою значною кількістю різношерстної інформації та даних досягти досить складно. Недотримання вимог безпеки й не використання сучасних та ефективних підходів до реалізації захищеної системи може загрожувати втратою часу, грошей та інших цінних ресурсів, а інколи й проблемами з законом [1].

Відповідно звіту RiskBased Security про тенденції порушення даних за 2021 рік, у 2020 та 2021 роках було втрачено 27,81 та 18,88 мільярдів записів, тоді як у 2019 році усього 4,681 мільярдів записів [1]. Попри те, що кількість таких випадків зменшилася у 2020 та 2021 роках, втрати даних стали більш масштабними. Також, ми можемо спостерігати відсутність кореляції між кількістю прецидентів втрати даних та кількістю втрачених даних. Це говорить про те, що навіть одне малозначне порушення безпеки може спричинити гірки

наслідки, котрі призведуть до значних проблем у системі. Щоб на практиці зменшити усі можливі ризики, потрібно ретельно пропрацювати усі потенційні загрози та ідеально дотримуватися усіх стандартів безпеки.

Найбільший удар у 2021 році отримали сфера охорони здоров'я, фінансів й страхування, інформаційна індустрія, наукова галузь, промисловість та галузь державного управління, котрі являють собою значну долю від усіх зломів [1]. Попри те, що у даному дослідженні розкриваються лише проблеми, що зачіпають загрози та вразливості веб-додатків, джерела даних випадків витоку даних можуть бути абсолютно різними.

Об'єктом дослідження є процес контролю якості та пошук вразливостей у сучасних клієнт-серверних веб-додатках.

Предметом дослідження є методи та способи контролю якості та пошуку вразливостей у сучасних клієнт-серверних веб-додатках.

Метою роботи є аналіз ефективних методів контролю якості та пошуку вразливостей у сучасних клієнт-серверних веб-додатках, а також вироблення практичних рекомендацій щодо вибору певного інструменту виявлення вразливостей, на основі результатів сканування системи й порівняння результатів проведеного виявлення загроз.

Основними завданнями при виконанні дипломної роботи можуть вважатися наступні:

- розгляд понять веб-додатків, побудови їхньох системи та їх клієнт-серверної архітектури, сучасних вразливостей веб-додатків та їх класифікації, методів контролю якості веб-систем в цілому та інструментів виявлення вразливостей зокрема;
- проведення класифікації інструментів сканування веб-додатків, розгляд специфіки їх роботи та особливостей;
- аналіз вразливого веб-додатку завдяки використанню інструментів сканування, котрі було розглянуто раніше та отримання результатів сканування у вигляді кількості виявлених загроз;

- визначення обґрунтованих висновків, вироблення та надання практичних рекомендацій, щодо доцільності застосування інструментів виявлення вразливостей веб-додатків вцілому та деяких зі сканерів зокрема.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У цьому розділі розглядаються сучасні способи контролю якості розроблюваних веб-систем. Особлива увага приділяється веб-додаткам, їх типам, їх архітектурній моделі, найпоширенішим критичним вразливостям веб-систем та їх класифікації, методам контролю якості й забезпечення безпеки розроблюваних веб-додатків, технологіям пошуку вразливостей й недоліків програмю, принципам роботи систем пошуку вразливостей у веб-додатках та деяким їх характерним особливостям.

1.1 Веб-додатки та їх суть

Веб-сайт являє собою деяку совокупність тісно пов’язаних між собою веб-сторінок, котрі мають єдине доменне ім’я та представляють доступ до деякого спільного за змістом контенту. Такий ресурс може бути розробленим будь-якою фізичною особою або приватною організацією для досягнення деяких цілей. Розташовується веб-сайт на одному або більше серверах та є доступним для користувачів через мережу Інтернет за допомогою веб-браузера.

У свою чергу веб-додаток відрізняється від звичного користувачам веб-сайту тим, що в ньому розробник робить акцент не на контент, котрий буде розміщено, а користувачів й їх взаємодію з веб-додатком. Це є однією з причин, чому веб-додатки мають складніші функціонал й архітектуру та потребують значно більшої кількості часу для їх реалізації, аніж звичайні веб-сайти. Такі застосунки працюють в умовах підключення до бази даних (БД), через надсилання запитів до БД та отримання від неї відповідних результатів [2].

Інтерфейс веб-додатків зазвичай реалізується за допомогою таких інструментів як: HTML (HyperText Markup Language), CSS (Cascading Style Sheets), JavaScript та деяких їх бібліотек, котрі дозволяють розробляти програмні рішення набагато швидке та комфортніше, уникаючи великої кількості проблем, пов’язаних з розробкою продуктів. Серед таких бібліотек можна виділити AngularJS та ReactJS, котрі гарно закріпилися у ніші розробки веб-додатків та є

незмінно найпопулярнішими, найгнучкішими й найоновленішими інструментами. Серверна частина програмних рішень відповідає за обробку різноманітнітних сценаріїв та дій що можуть бути виконані користувачем. Серед інструментів реалізації серверної частини мережевих програмних рішень є багато інструментів. Найпопулярнішими є інструменти на базі мов програмування JavaScript, Java, Python та відповідні бібліотеки, котрі спрощують розробку й, можна сказати, розширяють можливості розробників.

Серед основних характеристик веб-програм можна виділити наступні [3]:

- Можливість легкого проведення тестування програмного рішення завдяки різним видам тестування, серед яких є мануальне та автоматизоване види тестування.
- Веб-додатки не потрібно встановлювати на пристрої, що забезпечує таким програмам неперевершену кросплатформенність та дозволяє використовувати один і той же продукт на різних пристроях, що в свою чергу допомагає економити час та ресурси розробників.
- Модульність дозволяє легше розробляти й змінювати програму, підтримувати чистоту програмного коду, робити бекапи та вивантажувати дані на сервер.
- Веб-додатки розміщуються на хмарному хостингу, що надає їм гарну масштабованість.

1.2 Архітектура веб-додатків

В даному розділі представлена фундаментальна архітектура веб-додатків, а саме клієнт-серверна архітектура. Вона поділяється на різнорівневі архітектурні рішення, серед яких було виділено: одноврівневе, дворівневе, трирівневе та N-рівневе архітектурні рішення [4].

1.1.1 Клієнт-серверна модель

Клієнт-серверна модель веб-додатків являє собою загальну архітектуру більшості сучасних розроблюваних веб-додатків. Такі мережеві протоколи, як HTTP та HTTPS (HyperText Transfer Protocol Secure) повсякденно

використовують модель мережевого зв’язку клієнт-сервер для забезпечення своєї ефективної й коректної роботи [5]. Відносини між клієнтом та сервером здійснюються за протоколами TCP (Transmission Control Protocol).

Архітектурне рішення розподіляє веб-програму на дві логічно розподілених частини, котрі виконують різні задачі, задля забезпечення функціонування застосунку. Такі частини називають клієнтом й сервером. Клієнтська частина відповідає, насамперед, за користувацький інтерфейс застосунку та ініціює запити на сервер. Веб-браузер є прикладом клієнту. Серверна частина займається обробкою запитів користувачів, котрі надсилаються з клієнту, й надсилає відповіді на запити назад на клієнт, іншими словами, надаючи клієнту необхідні йому ресурсі й дані [6]. Сервери часто знаходяться на відокремлених комп’ютерах, тому контакт з ними відбувається через мережу.

На рисунку 1.1 зображена взаємодія клієнт-серверної архітектури.

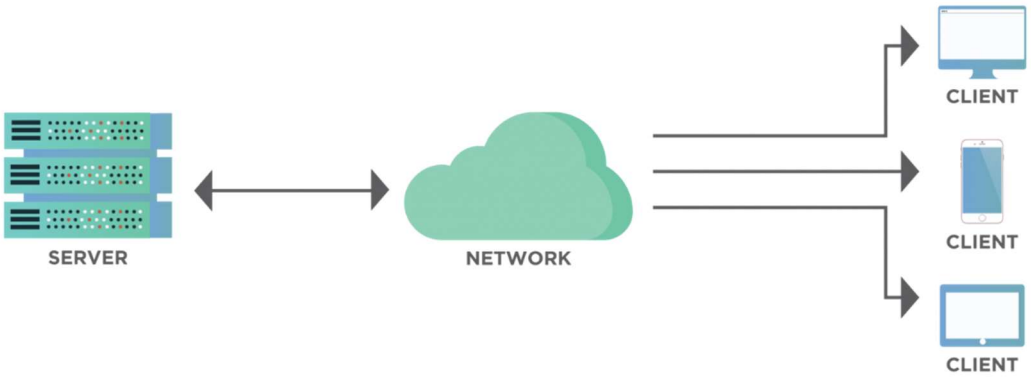

Рисунок 1.1 – Взаємодія серверу та клієнтів через мережу

Даний цикл зв’язку між двома комп’ютерами продовжується, доки залишається мережеве з’єднання між ними. Майже завжди сервер контактує з деякою БД, маніпулюючи деякими користувацькими даними. Іншими словами, сервер перевіряє БД на наявність необхідних клієнту даних та повертає їх на клієнтську частину застосунку у тому разі, коли така необхідність існує й запитувані дані було знайдено. Сервер додає, змінює та видаляє дані БД, тим самим забезпечуючи роботу всієї системи. Сервер має змогу виконувати роль клієнта, у тому випадку, якщо він запитує дані у БД своїми силами. Клієнт-

серверна архітектура надає можливість змінювати БД декільком користувачам через графічний інтерфейс [7].

На рисунку 1.2 зображено типову архітектуру веб-додатків.

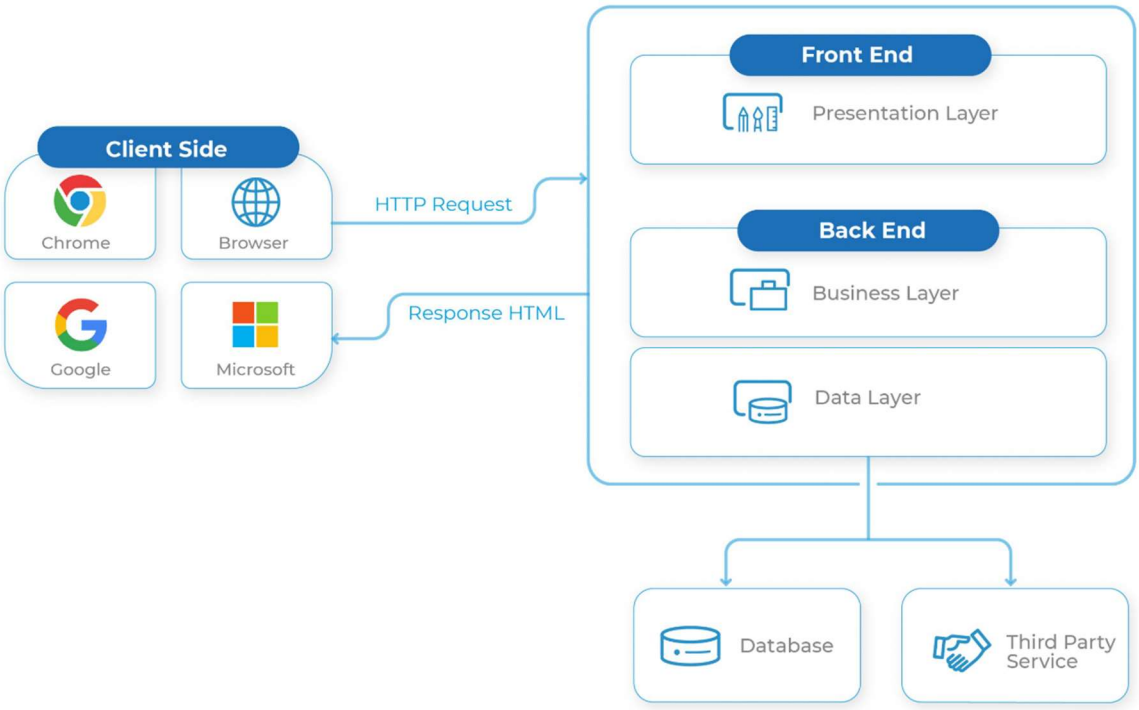

Рисунок 1.2 – Типова архітектура веб-додатку

1.1.2 Типи клієнт-серверної архітектури

Клієнт-серверна архітектура поділяється на різноманітні види, беручи собі за основу бізнес-логіку, котра створена для запиту, який в цей час виступає посередником між клієнтом і сервером. Далі буде розглянуто різноманітні типи прикладів такої архітектурної моделі.

Розглянемо однорівневу архітектурну модель клієнт-серверу, за якою усі пакети, які стосуються програми, групуються та виступають як єдине ціле. Бізнес-логіка, користувацький інтерфейс, логіка БД та БД в цілому відіграють одну сутність.

Однорівнева архітектура має різноманітні служби, які вона спроможна запропонувати розробникам [4]. Дані служби роблять її надійним джерелом в очах користувачів, але має такі недоліки, як проблеми зі складністю в управлінні. Розходження даних та дублювання роботи є частими неприємними проблемами такого архітектурного рішення. У однорівневій архітектурі існує декілька рівнів

представлення, таких як: презентаційний, бізнес-рівень та данні, котрі поєднуються шляхом виняткового програмного пакету. Даний рівень архітектури переважно зберігає дані в локальних системах чи на загальному сховищу.

На рисунку 1.3 зображено однорівневу архітектуру клієнт-серверної моделі.

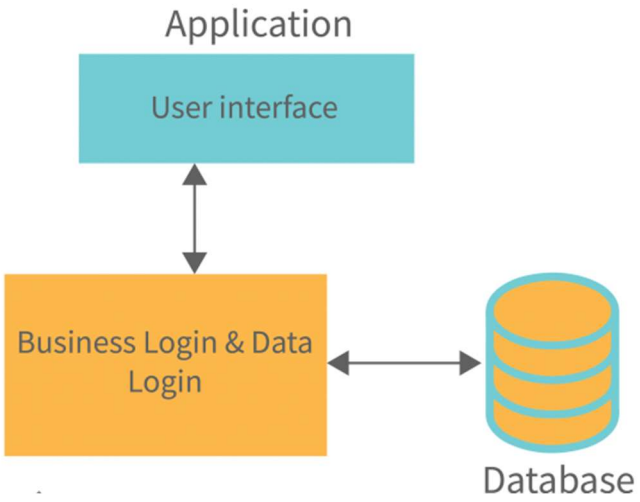

Рисунок 1.3 – Однорівнева архітектура клієнт-серверної моделі

Далі розглянемо дворівневу клієнт-серверну архітектуру. Вся сутність такого архітектурного рішення поділяється на дві частини або, іншими словами, рівні. Основна БД виступає як окрема частина в даній архітектурі. Розробка БД відбувається окремо, основна частини ПЗ має таку логіку, яка пов’язана зі користувацьким інтерфейсом, та бізнес-логіку й логіку БД, щоб мати зв’язок з БД та маги змогу обробляти програми. Дане архітектурне рішення має значні переваги за однорівневу архітектуру, завдяки кращого середовища [4].

Після проведення порівняння однорівневої та дворівневої архітектур, можна зробити висновок, що дворівневі більш швидші, причиною цьому являється відсутність посередника між клієнтом й сервером. Дворівнева система надає спроможність уникати мішанини клієнта.

На рисунку 1.4 зображено дворівневу архітектуру клієнт-серверної моделі.

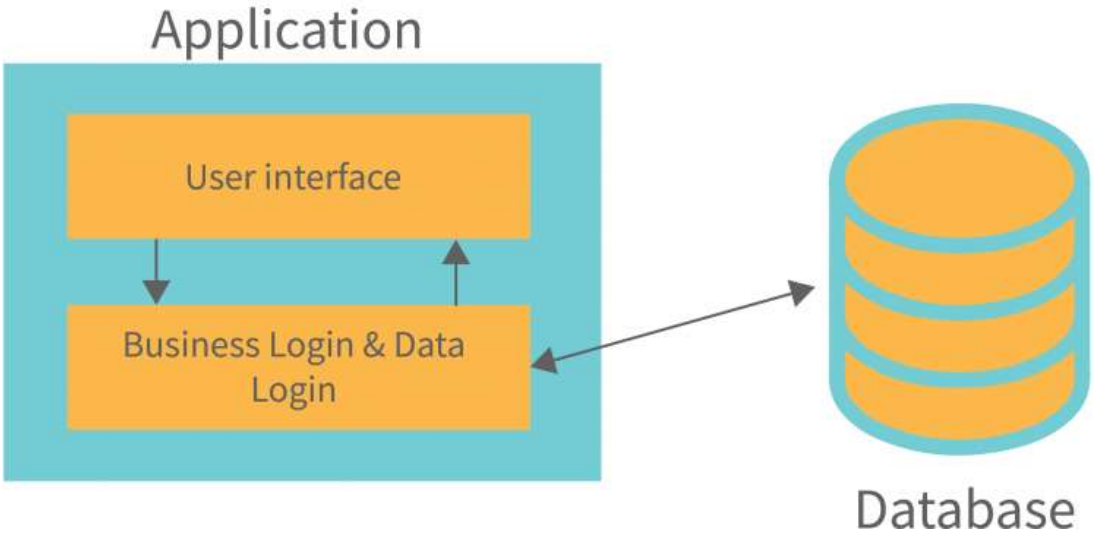

Рисунок 1.4 – Дворівнева архітектура клієнт-серверної моделі

Трирівнева система включає в себе проміжне ПЗ між клієнтом та сервером. В цей же час дворівнева система, не має проміжного ПЗ. При запиті клієнтом інформації від сервера, запит спершу побачить проміжне ПЗ. Далі шлях запиту надсилається до серверу, щоб пройти обробку. Тим же шляхом сервер відправить відповідь клієнтку на його запит.

Трирівнева архітектура складається з трьох основних рівнів, таких як: презентаційний рівень, прикладний рівень та рівень БД [4]. Різні кінці кожного рівню мають контроль над одним. Клієнтські прилади мають контроль над рівнем презентації, у той же час проміжне ПЗ та сервер беруть контроль над рівнем додатків та рівнем БД відповідно. Трирівнева архітектура вважається більш безпечною, в ній присутні невидимі структури БД та забезпечується надання відповідної цілісності даних.

На рисунку 1.5 представлено трирівневу клієнт-серверну архітектуру.

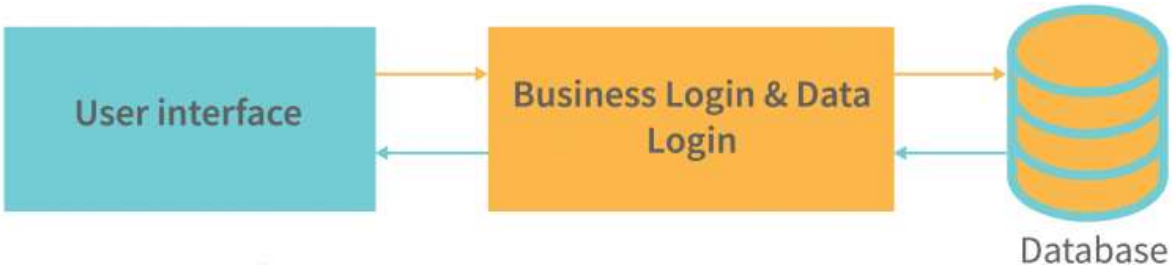

Рисунок 1.5 – Трирівнева клієнт-серверна архітектура

Переваги клієнт-серверної архітектури [8]:

1) Легке управління – процес управління файлами досить легкий, тому що всі файли є на одному сервері. У клієнт-серверних мережах присутнє найкраще управління для того, щоб відстежувати та шукати записи необхідних файлів.
2) Доступність – користувачі мають можливість входу до системи незалежно від місцезнаходження або вибраної платформи, яка надає можливість кожному робітнику мати доступ до своєї корпоративної інформації без підключення режиму терміналу чи процесора.
3) Кожен сервер масштабований – у клієнт-серверній мережі присутня висока масштабованість. Кожного разу, коли клієнт потребує, він спроможний додати ще більше ресурсів, наприклад клієнти та сервери, таким засобом збільшуючи обсяг серверу без перерв. Сервер централізований, тому доступ до мережевих ресурсів не може бути проблемою, для самої конфігурацій треба мало робітників, навіть коли розмір серверу збільшується.
4) Централізоване керування – у клієнт-серверний мережах присутні переваги централізованого управління, тому що вся присутня інформація знаходиться в одному місці. У адміністратора мережі присутній повний доступ та контроль над управлінням та адмініструванням системою. Будь-які проблеми, котрі можуть або виникають у мережі, дозволено вирішувати з одного визначеного централізованого місця. Результатом цього стало легкість у оновленні даних та ресурсів.
5) Захищеність – централізована архітектура клієнт-серверу мережі дуже добре захищає дані. Створену резервну копію можливо використовувати щоб відновити усі втрачені дані та файли, як, наприклад, нав’язування облікових даних, таких як ім’я користувача й

паролі. Також є інший спосіб забезпечення контролю та доступу полягає у нав'язуванні облікових даних, як ім'я користувача або пароль.

Недоліки клієнт-серверної архітектури [8]:

1) Надійність – централізована природа клієнт-серверних мереж при збої чи перешкодах на головному сервері робота мережі буде зупинена. Це дає нам зрозуміти що, клієнт-серверні мережі мають не найкращу надійність.
2) Необхідність у регулярному технічному обслуговуванні – сервери працюють у мережі день у день. Це дає нам зрозуміти, що вони потребують постійне та якісне обслуговування. При проблемах має бути забезпечено миттєве усування причин їх виникнення й усунення з'явившихся недоліків. Потрібне постійне призначення менеджеру мережі для того, щоб обслуговувати сервер.
3) Потребує витрат – ціна налаштування та обслуговування сервера дуже висока, так як і мережеві операції. Мережі дуже потужні, та її придбання не може бути дешевим. Тому не кожен користувач може дозволити ними скористатися.
4) Перевантаженість мережі – при великому напливі користувачів, можуть з'являтися такі проблеми, як збої або уповільнення з'єднання. Перевантажений сервер є причинної проблем при запиті до інформації.

1.2 Сучасні загрози безпеці веб-систем

Вразливі місця системи являють собою недоліки або слабкі місця програми, котрі можуть буди обхідним шляхом для хакера, щоб призвести до використання вразливостей та порушення безпеки системи. Через глобальність Інтернету, веб-застосунки особливо вразливі до кібератак, які мають змогу надходити абсолютно з різних місць та різних напрямків. Постійний контроль над вразливістями веб-застосунків та методами тестування безпеки таких систем є особливо важливими елементами для забезпечення безпеки веб-додатків [9].

1.2.1 Поточний стан безпеки сучасних веб-застосунків

Атаки на веб-застосунки являють собою найнебезпечнішу загрозу безпеці багатьох сучасних організацій. Атаки спрямовані на веб-застосунки становили 40% від усіх зламів у 2015 році й вже понад 50% у 2021 році [10]. Згідно з аналізу WhiteHat Security на момент 2021 року, так чи інакше, принаймні 50 відсотків веб-застосунків у таких сферах, як: освіта, промисловість, роздрібна торгівля, комунальні служби та охорона здоров'я мають принаймні одну вразливість, котра можливо використати для зламу. В ході дослідження було виявлено, що через те, що з кожним роком все більша кількість галузей націлюються на розвиток у мережі, ризики витоку даних залишаються високими, що позначається на загальній безпеці та збільшенні вразливості. Також було виявлено, що лише деякі з найпоширеніших ризиків від WhiteHat можна побачити в списку OWASP (Open Web Application Security Project), в котрому регулярно оновлюються найбільш розповсюджені вразливості.

Знаходження проблеми та виявлення вразливості не вирішує проблему безпеки саме по собі. Щоб виправити виявлені вразливості треба приділити деякий час. Час на виправлення важливих критичних застосунків сягає в середньому 194 дні. Середній час відкриття критичних вразливостей становить 300 днів і більше, а вразливостей с високим рівнем ризику понад 500 днів. Чим більша кількість днів, тим більші шанси злодіїв на добування користі від вразливості. Також, великий відсоток веб-застосунків залишається вразливим завжди, що каже про те, що розробники не в змозі впоратися з вразливостями своїх продуктів.

Усунення проблем безпеки та виявлення вразливостей ще на етапі проектування й розробки веб-застосунку потребує значно менших ресурсів, аніж їх виправлення у вже існуючому продукті. У існуючому застосунку процес виправлення вразливостей здатний призвести до втрати працездатності веб-застосунку, що, в першу чергу, несе втрату репутації, коштів й ресурсів з ціллю вирішення виникшої проблеми. Вдосконалення вже існуючого веб-застосунку у

більш захищене рішення може коштувати значно більше сил і коштів, аніж його початкова реалізація у вигляді захищеного застосунку. Прикладом може слугувати слабка система паролів, у разі модифікації якої усі користувачі повинні будуть змінити та оновити паролі до своїх облікових записів. У разі дотримання усіх сучасних стандартів безпеки веб-застосунку, такої ситуації можна запросто уникнути.

На сьогоднішній день, розробка веб-застосунків, беручи до уваги стандарти безпеки, є неодмінним обов'язком. Безпеці треба приділяти увагу на рівні з питанням функціональності веб-застосунку. Список загроз та вразливостей постійно розвивається, не стоячи на місці, тому розробники мають розбиратися у сфері безпеки, щоб залишати свій продукт безпечним та захищеним. На жаль, впоратися з цією задачею буває не легко, тому у якості допомоги розробники можуть звертатися до стандартів безпеки або користуватися іншими інструментами, котрі здатні допомогти у забезпеченні захисту їхніх програмних рішень.

1.2.2 Критичні вразливості веб-додатків

Проблеми, що пов'язані з сучасною безпекою та актуальною захищеністю веб-додатків зростають з кожним роком, що пов'язано напряму з їх масштабованістю й складністю. З цією причини виправлення всіх недоліків захисту здатне знизити продуктивність програмних рішень.

Опираючись на звіт Invicti AppSec 2021 року, половина робочого процесу розробників та тестувальників веб-застосунків використовується не для подальшої розробки веб-продуктів, а для вирішення вже існуючих проблем їх безпеки. Проблеми ускладнюється тим, що багато з вже реалізованих програмних рішень були розроблені вже давно та й близько не відповідають сучасним стандартам безпеки, що масштабує можливості злому та атаки на застосунки.

Таким чином, третина проблем захищеності опиняється у розробці, що напряму впливає на час котрий буде затрачена на їх виправлення. Саме тому,

застосування сучасних загальноприйнятих методів проектування захищеного продукту є першочерговою задачею, що має бути виконане не лише на перших етапах проектування застосунку, а й підтримуватися на кожній стадії життєвого циклу розробки ПЗ.

Вразливості веб-систем з'являються через деякі системні проблеми й слабкості, використовуючи які зловмисник може отримати доступ до тих засобів ПЗ, до яких користувач не має доступу за задумкою. Часто такі слабкості здатні існувати роками з моменту реалізації продукту.

Серед можливих причин виникнення вразливостей можна виділити погано продуманий дизайн застосунку, відсутність перевірки вхідних даних та очищення форм вводу користувацьких даних та некоректно відлажені сервери, що в свою чергу призводить ко компроментації застосунку та безпеки даних користувачів. Такі проблеми з'являються, тому що такі ПЗ мають працювати у мережі та підтримувати роботу застосунку через мережу для багатьох користувачів одночасно.

Стандарти безпеки веб-програм регулюються відповідними органами, що розроблюють та досліджують стандарти, забезпечують та допомагають підтримувати захищеність програмних веб-рішень, виявляти та усувати недоліки безпеки складних систем. Найкритичніші вразливості сучасних веб-застосунків та способи підтримки актуальної безпеки веб-продуктів описані в списку OWASP Top Ten 2021 року.

OWASP являє собою головний ресурс з інформацією про стандарти, відомі методи захисту та інструменти безпеки. Через велику базу користувачів та відкриту спільноту, OWASP роками залишається своєрідною бібліотекою, в котрій містяться різноманітні документи, що допомагають вирішити проблеми безпеки при розробці продуктів.

Головною метою проекту OWASP Top Ten є формулювання головних загроз безпеки, визначення найважливіших та найбільш критичних недоліків захищеності веб-застосунків. Даний список створено за сприяння об'єднаної

спільноти проекту. На початку існування ціллю проекту було покращити знання розробників, щодо безпеки, але згодом проект по суті став справжнім стандартом безпеки веб-застосунків. Наразі актуальною є версія 2021 року, котра вийшла відносно недавно й включає в себе найбільш актуальний список загроз та вразливостей.

Останню версію списку OWASP складено на базі опитувань у відповідній галузі та більш ніж 40 відгуків від компаній, що займаються забезпечення безпеки. Також, була зібрана інформація з більш ніж ста тисяч застосунків та API (Application Programming Interface), що дозволило створити досить обширний інформаційний фундамент задля подальшого аналізу даних.

Розподіл за принципами критичності вразливостей веб-додатків тісно залежить від перспективи використання недоліків системи, можливості виявлення вразливостей та їх безпосередньої загрози сучаснім системам веб-застосунків. Найнебезпечними загрозами котрі можуть бути виявлені або використані є ті, що ініціюють повне припинення коректної роботи системи, але в той же час існують і такі, котрі взагалі не завдають ніякої шкоди системам та не мають на неї жодного впливу.

На рисунку 1.6 зображено актуальний рейтинг найкритичніших веб-загроз та наглядні зміни у порівнянні зі списком головних загроз у 2017 році [11].

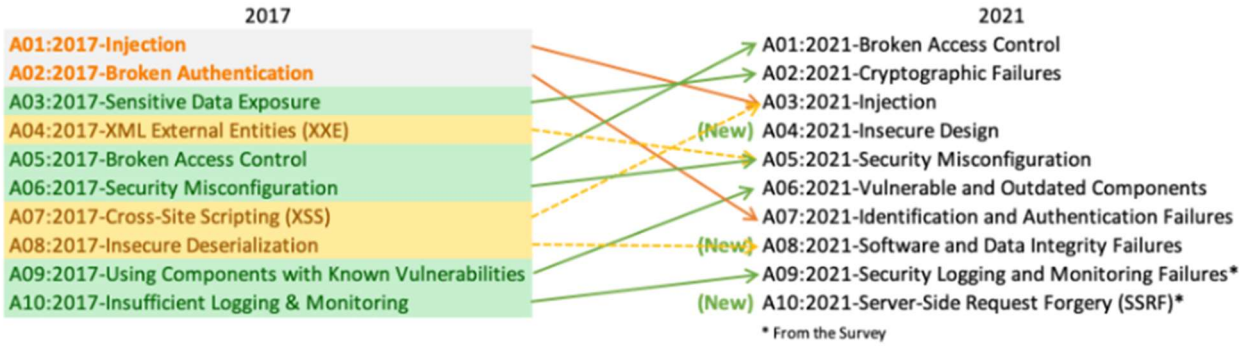

Рисунок 1.6 – Список найкритичніших сучасних веб-загроз у порівнянні з рейтингом 2017 року

Згідно з актуальним списком веб-загроз та вразливостей OWASP Top Ten у 2021 році найкритичнішими стали наступні [1]:

1) Порушений контроль доступу. В реалізованих веб-застосунках має бути реалізовано доступ до інформації згідно з привілегіями користувачів. У випадку ігнорування безпеки доступу до інформації, це може дозволити користувачам отримати доступ до важливої інформації, на доступ до якої вони не мають повноважень та може призвести до незворотних наслідків і втрати критично важливих даних.
2) Криптографічні збої. Криптографічні проблеми та несправності з'являються, якщо криптографічні методи не здатні забезпечити достатню безпеку. Звичайні складності криптографічного контролю охоплюють неактуальні криптографічні шифри та протоколи.
3) Ін'єкції. За допомогою ін'єкцій злочинці мають змогу впроваджувати вигідні їм шкідливі дані у веб-застосунок. Таким чином, в ін'єкціях можуть перебувати інструкції, котрі перенаправлятимуть дані, що приведе до отримання несанкціонованого доступу до них. Крім того, ін'єкції здатні власноруч змінювати веб-застосунок. До того ж, у новій версії списку OWASP 2021 року до цієї категорії було додано XSS (Cross-Site Scripting), тобто міжсайтовий сценарій.
4) Небезпечний дизайн. Критичні загрози та важливі проблеми безпеки можуть з'являтися через вади дизайну. Дизайн має обов'язково охоплювати шаблони проектування й безпечну еталонну архітектуру. Невміння визначити бізнес-ризики здатне призвести проблеми на проекті.
5) Неправильна конфігурація безпеки. Проблеми з конфігурацією безпеки відбуваються тоді, коли застосунки дають можливість на незаконний доступ до інформації та маніпулювання даними, у томи числі: модифікацію, зміну та крадіжку даних.

6) Вразливі та застарілі компоненти. Дана категорія вразливостей охоплює всі застарілі компоненти ПЗ, експлуатація яких може вийти боком у вигляді критичних ризиків безпеки системи. Регулярна профілактика безпеки таких компонентів у вигляді їх сканування дозволяє вияви-ти проблеми на ранньому етапі.
7) Помилки ідентифікації та автентифікації. Категорія охоплює вразливості, що пов’язані з проблемами автентифікації та ідентифікації. Такі проблеми включають в себе фальсифікацію облікових даних та "brute force" атаки. Вразливості цієї категорії часто з’являються через брак багатофакторної автентифікації та перевірки сеансу користувача.
8) Порушення цілісності даних зокрема та ПЗ в цілому. Категорія охоплює припущення зроблені щодо критичних даних без перевірки цілісності, на рахунок оновлень ПЗ та конвеєрів CI/CD (Continuous Integration/Continuous Delivery). Також, до категорії включена незахищена десеріалізація, котра має місце у тому разі, коли застосунки не десеріалізують вразливі об’єкти. Злочинці можуть використовувати дані, котрі приходять на сервер, щоб проводити атаки.
9) Помилки в моніторингу на логуванні проблем безпеки. Дана категорія вразливостей включає в себе такі вразливості, котрі пов’язані з незадовільним моніторингом та логуванням збоїв. Такі вразливості можуть призводити до проблем зі спроможністю реагувати на збої у роботі застосунку. Логування та моніторинг є головними рішеннями задля попередження та виявлення порушень безпеки.
10) Підробка запитів на стороні сервера. Вразливості даної категорії надають зловмиснику можливість реалізувати надсилання сервером веб-застосунку запитів до обраного зловмисником домену. Така вразливість може виникнути у разі, якщо веб-застосунок отримує дані без перевірки адреси, котру надає користувач. Також, зловмисники

здатні змусити сервер надіслати підроблений запит у несанкціоноване місце.

1.3 Методи контролю якості та безпеки веб-додатків

Програмісти й інші користувачі використовують різноманітні види тестування, чим не допускають вразливостей й дозволяють системам залишатися захищеними. Для захисту складових частин системи й користувацьких даних треба застосовувати не тільки популярні методи проведення тестування веб-додатків, а й дотримуватися безпечних практик написання коду ПЗ [12].

Гарним стартом для початку забезпечення безпеки є end-to-end та unit тестування, котрі тестують основні складові частини ПЗ, застосовуючи автоматизоване тестування. Серед мінусів такого методу тестування можна виділити відсутність перевірки тих вад системи, котрі можуть призводити до розкриття даних [13]. Функціональне тестування приділяє особливу увагу забезпеченню виконання головних цілей ПЗ.

Також, в світі наявні інші способи аналізу, котрі асимілюють проблеми, що були не вирішені після використання типових методів тестування ПЗ. Тоді на допомогу приходять методи статичного тестування безпеки програми, що називається SAST (Static Application Security Testing) та динамічне тестування безпеки програми, що називається DAST (Dynamic Application Security Testing). Далі ознайомимося з принципами роботи кожного з них для того, щоб мати краще розуміння суті їх роботи.

1.3.1 Підхід SAST

Статичне тестування безпеки додатка (SAST) є методом, з використанням якого тестувальники ПЗ мають можливість дослідити програмний код системи для знаходження критичних недоліків й прогалин системи безпеки [14].

Інструменти, що працюють на базі SAST виконують свою роботу за принципами «білої скриньки», скануючи складові частини ПЗ, для виявлення вразливостей. Такі інструменти використовуються починаючи з початкових етапів розробки ПЗ й до того моменту, коли розробка продукту вже буде

закінчена. Дану процедуру можливо використати після поєднання усіх модулів ПЗ в деякому середовищі. Присутня можливість автоматизування процесу, що може суттєво заощадити людські ресурси.

Методологія SAST шукає проблеми безпеки програмного коду ПЗ ще на початкових стадіях розробки й не створює нові недоліки системи безпеки на пізніх етапах створення продукту.

Основними принципами роботи інструментів SAST є наступні [14]:

1) Інструмент SAST виконує аналіз програмного коду ПЗ, без необхідності використання такого ПЗ, для того, щоб мати ґрунтовне представлення складових частин, модулів й налаштувань.
2) Інструмент виконує перевірку коду програми переглядає кожен рядок з вверху вниз та інструкцію за інструкцією, виконуючи порівняння їх за встановленими вказівками. Він виконає перевірку вашого вихідного коду, для того, щоб знайти вразливості та недоліки, таких як впровадження SQL (Structured Query Language), переповнення буфера, проблеми XSS та інші.

Даний метод проведення аналізу й виявлення проблем, здатний посприяти створенню плану дій, для їх корегування й зміцнення безпеку ПЗ. Важливо пам'ятати, що інструменти SAST спроможні в якості результати своєї роботи надавати хибні спрацьовування. З цією причини, користувачі повинні гарно ладити з розробкою системи безпеки й архітектури ПЗ, щоб знаходити такі хибні спрацьовування. Також існує варіант внесення корегувань до програмного коду, для того щоб не допустити помилкових спрацьовувань й знизити їх кількість.

1.3.2 Підхід DAST

Динамічне тестування безпеки (DAST) являє собою відмінний від статичного метод тестування й вдається до застосування підходу «чорної скриньки», допускаючи, що люди, котрі виконуватимуть тестування не можуть не отримати доступу до програмного коду ПЗ чи його складових частин [15].

Аналіз программи за методом DAST проходить ззовні, для цього застосовуються актуальні дані.

Динамічне тестування ставить своєю ціллю слідкувати за тим, як поводиться ПЗ згідно з проведеними атаками й дозволяє знаходити вразливості, котрі можна використати у системі. Для застосування підходу DAST перш за все необхідно розпочати роботу ПЗ, щоб отримати можливість зв'язуватися й взаємодіяти з нею й застосовувати певний набір атак й експлойтів.

Використання інструментів забезпечення якості, що працюють за методом DAST на всіх етапах життєвого циклу ПЗ дозволить виявляти недоліки системи захисту й запобігати багатьом проблемам, серед яких можна виділити витік конфіденційних даних.

Також, важливо помітити, що DAST можливо використовувати у мануальному форматі чи застосовуючи сканери DAST, котрі надають можливість проведення XSS атак, атак з використанням SQL ін'єкцій та інших. Також, сканери DAST спроможні виявляти проблеми, що пов'язані з процесом автентифікації, налаштуванням серверів й нестійким шифруванням даних.

Інструменти DAST прості у використані, вони виконують безліч складних речей за лаштунками, для того, щоб спростити тестування [15]:

1) DAST інструменти направлені на збір даних про ПЗ. Програма сканує кожну сторінку коду та отримує вхідні дані, для того, щоб збільшити можливу площу атаки.
2) Далі йде активне сканування ПЗ. DAST інструмент відправляє різноманітні вектори атак до кінцевих точок, які були знайдені раніше, для того, щоб знайти уразливості, як-от XSS, SSRF (Server-side request forgery), ін'єкції SQL тощо. Також, безліч інструментів DAST надають можливість створювати сіої власні сценарії атак, для того, щоб перевіряти наявність нових проблем.
3) При завершенні цього циклу інструмент покаже результати. Якщо була знайдена вразливість, він негайно надасть інформацію про вразливість,

її тип, URL (Uniform Resource Locator) адресу, серйозність, вектор атаки та допоможе усунути знайдені проблеми.

Інструменти контролю якості, що працюють за методом DAST прекрасно знаходять проблеми автентифікації та налаштувань тоді, коли здійснюють вхід до ПЗ. Вони забезпечують використання деяких вхідних даних, для здійснення спроб зламу системи. Далі інструмент здійснює порівняння отриманого результату з бажаним результатом, для виявлення загроз.

1.3.3 Сканери вразливостей веб-додатків

Сканер веб-застосунків являє собою таке ПЗ, котре робить аналіз системи деякими автоматизованими засобами тестування веб-застунків, використовуючи метод DAST й виявляє проблемні місця системи безпеки ПЗ. Інструменти виявлення загроз існують для того, щоб здійснювати аналіз якості створених систем й запобігати загрозам, котрі спроможні бути застосовані хакерами для отримання несанкціонованого доступу до системи й конфіденційних даних протягом усього життєвого циклу розробки ПЗ. В ході аналізу веб-застосунків здійснюється дослідження програмного коду ПЗ, на предмет незахищеного коду, котрий являє собою вразливості системи, в тому числі ті, котрі знаходяться в списку вразливостей OWASP.

У сканерів відсутній доступ до вихідного коду й вони можуть тільки виконувати функціональне тестування та стараються віднайти вразливі місця безпеки. Сканери вразливостей користуються переліком БД, котрий постійно оновлюється, щоб знаходити й розділяти за категоріями критичності проблеми безпеки систем й визначати важливість й критичність виправлення таких недоліків захисту. Окремі програми аналізу виконують свою роботу настільки глибоко, що автоматично знаходять та вирішують проблему, таким чином зменшуючи об’єм роботи розробників ПЗ й тестувальників механізмів захисту. На ринку контролю якості й безпеки веб-застосунків існують як безкоштовні сканери, так и платні програмні рішення [16].

ПЗ для аналізу якості веб-додатків застосовують, щоб знаходити й запобігати загрозам додатків, котрі з'являються через накоректне налаштування системи чи погано написаний програмний код. Новітні рішення надають можливість виконувати аналіз як використовуючи автентифікацію, так і в безавтентифікаційному режимі. Такі інструменти працюють за принципами SaaS (програмне забезпечення як послуга), пропонуючи роботу через мереже підключення й надається в якості ПЗ [16]. Кращі сканери спроможні створювати звіти з результатами знайдених проблем, з наданням спроможності налаштування встановленого ПЗ, хостів, портів й сертифікатів.

Сканер спершу виконує аналіз усієї системи, при цьому грунтовно скануючи всі файли й наглядно представляючи всі структурні особливості системи. Наступним кроком виконання стають спроби проведення атак на системи з логуванням отриманих результатів пошуку загроз й пропонованими рішеннями проблем [17]. Порівнюючи інструменти SAST з інструментами виявлення вразливостей методом DAST, останні не можуть отримати фактичного доступу до програмного коду аналізованої системи й здатні тільки виконувати атаки на системи, намагаючись використати поширені вразливості таких систем, котрі містяться у БД.

Автентифіковане сканування надає можливість інструменту проведення аналізу здобувати доступ безпосередньо до мережевих активів завдяки протоколам віддаленого адміністрування, серед яких можна виділити безпечну оболонку SSH (Secure Shell) чи протокол віддаленого робочого столу RDP (Remote Desktop Protocol), та автентифікуватися завдяки облікових даних [18].

Неавтентифіковане сканування являє собою підхід до аналізу системи, користуючись яким можна запросто отримати значну кількість некоректних результатів й не спроможне отримати повні дані про активну ОС та ПЗ що інстальоване й використовується. Такий підхід, як правило, використовують розробники й тестувальники, котрі намагаються встановити поточне становище зовнішньої системи безпеки [18].

Інструменти виявлення вразливостей мають за собою деякі межі, серед яких можна назвати наступні:

1) Проаналізовані й відповідно протестовані сканери не мають можливості визначаті всі недоліки безпеки системи, котрі містить програма. Всі вони мають недоліки виявлення деяких видів вразливостей. Одночасно сканери можуть бути ефективними рішеннями у виявленні одних категорій вразливостей і в той же час мати вагомі недоліки у виявленні вразливостей деяких інших категорій. Наприклад, використовувані нами сканери виявилися недостатньо ефективний й точними у виявленні та визначенні таких категорій недоліків системи веб-додатку як: Криптографічні збої та Порушення цілісності ПЗ та даних.
2) Використовуючи розглянуті інструменти виявлення вразливостей може бути тяжкою задачею виявити проблеми контролю доступу, неявні гарно замасковані обходи автентифікації та атаки з використанням декількох прикриттів процесу атаки на систему веб-додатку.
3) Процес автоматичного виявлення загроз має сканувати систему на предмет усіх загроз, які інструмент здатний виявити, що займає значно більшу кількість часу й порівняно неефективно з тими інструментами, котрі мають змогу виконувати більш цільові перевірки системи на предмет виявлення конкретних вразливостей або категорій вразливостей.

2 РОЗГЛЯД ІНСТРУМЕНТІВ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ

Серед актуальних рішень щодо оцінки веб-додатків на рахунок властивих їм вразливостей запропоновані сканери веб-систем з відкритим кодом.

Далі будуть представлені головні аспекти, того, яким має бути інструмент сканування вразливостей веб-додатків, щоб виконувати свою роботу ефективно:

1) Сканери повинні включати в себе протоколи та алгоритми автентифікації, котрими користуються веб-застосунки.
2) Методи відправлення вхідної інформації з включеною в систему функцією пошуку проблемних сторін веб-застосунку з малим ступінем помилок повинні підримуватися системою.
3) Система повинна бути зрозуміла для її користувача.
4) Сканер повинен мати постійні оновлення системи безпеки.
5) Якщо обирати інструмент серед платних сканерів, то обране ПЗ повинне повністю та цілком відповідати своїй ціні.

Дане досліжнення проводиться задля того, шоб провести актуальну оцінку можливостей інстрементів виявлення вразливостей веб-додатків в умовах сучасних веб-систем й критичних загроз таким системам. Щоб краще оцінити ефективність роботи сканерів й покращити варіативність виявлення, сканери можна оцінювати за категоріями найпоширеніших вразливостей OWASP Top Ten.

Далі наведено короткий опис відомих інструментів оцінювання веб-додатків.

2.1 OWASP ZAP

В системі проєктів OWASP, попитом найбільш користується OWASP ZAP (Zed Attack Proxy). Він являє собою кросплатформенний засіб пошуку проблемних сторін у системі. У порівнянні з системами конкурентами, даний інструмент активно приймає участь у тестуванні веб-застосунків. Інструмент має відкритий програмний код, скачати даний продукт можно безкоштовно. OWASP

ZAP доступний до використання такими людьми, як: розробники користувацького UI, розробники API, QA інженери й експерти з безпеки

Задля керування тестування за допомогою інструменту виявлення OWASP ZAP, користувачу запропоновано використовувати користувацький UI, або ж керувати програмою використовуючи інтерфейс командного рядку, тобто термінал програми [19].

На рисунку 2.1 представлено інтерфейс ZAP.

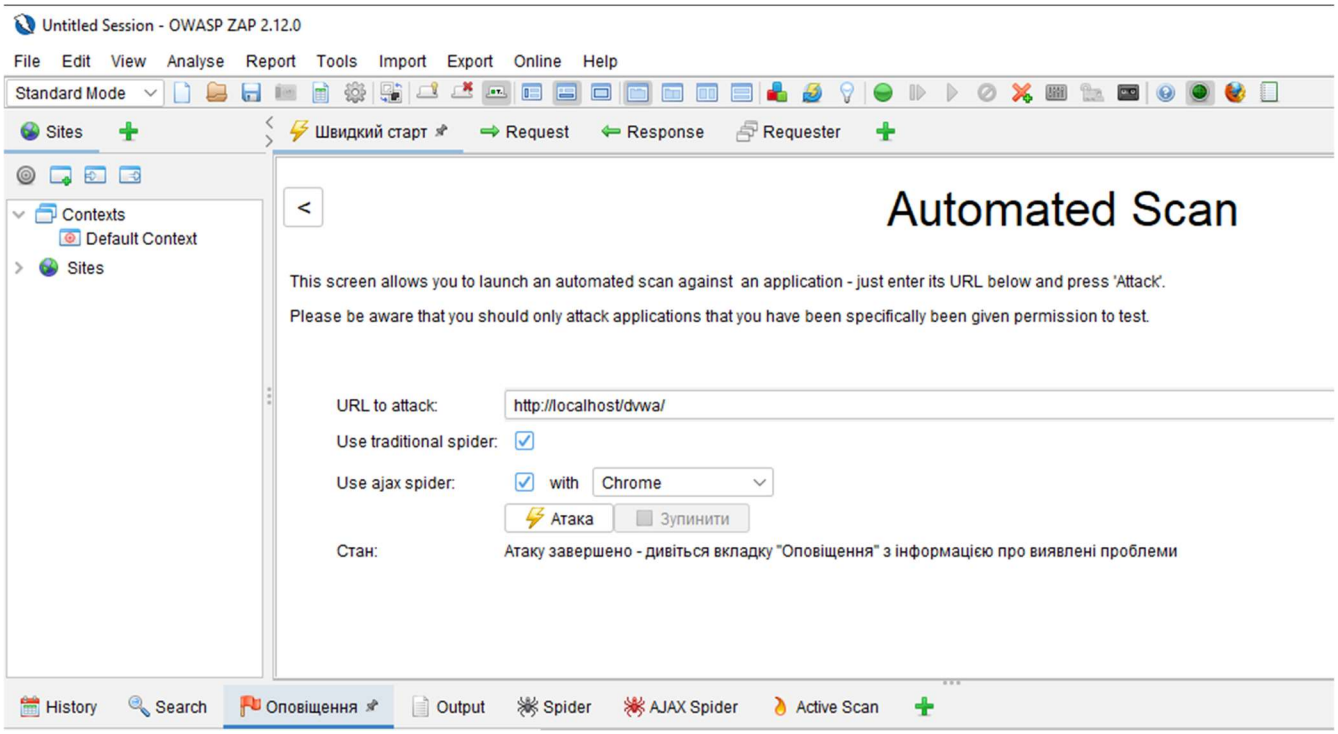

Рисунок 2.1 – Інтерфейс ZAP

Стилі ZAP доволі малі та короткі, які представленні з мізерної кількістю різноманітних метрик для пошуку параметрів і показників. В числі мінусів веб-застосунку можливо зазначити наявність активних типів, час пошуку й проблемні сторони. Інтуїтивно зрозумілий дизайн інструменту спрощує роботу з ним.

Загальні функції сканеру ZAP включають в себе наступні:

- Здобуття контролю над проксі-сервером дозволяє веб-браузеру здійснити конфігурування проксі-сервер, що дозволить спостерігати й маніпулювати усіма відправленими запитами й отриманими на них відповідями;

- Інструмент виявлення вразливостей ZAP надає користувачу можливість використовувати як методи проведення пасивного сканування, так і оцінювання проблем безпеки методом активного сканування. Оскільки тестування пасивними інструментами працює без спроб атакувати тестовану систему, то це робить їх надійними для застосування у виявлені вразливостей якого завгодно ПЗ. Тобто, пасивне сканування реалізовується завдяки тривалого часу роботи, оцінюючи запити й отримані на них відповіді, без можливості ідентифікувати конкретні типи знайдених вразливостей. В свою чергу, оцінювання вразливостей методом активного сканування забезпечує здійснення більш обширної кількості атак, але для того щоб його застосувати треба отримати формальне схвалення.
- Функція Spider має можливість бути застосована, щоб виконати аналіз системи, щоб знайти неявні веб-сторінки й загублені посилання.
- Метод аналізу «грубою силою» застосовується, щоб роздобути конфіденційні та чутливі дані. Використовуючи його інструмент ZAP має можливість виявити файли, на які немає прямих посилань, завдяки багатьом спробам;
- Функція фаззингу має змогу використовувати, щоб знаходити погано виявляємі вразливості, котрі не можуть знайти методи автоматичного аналізу;
- Функція автоматичного виявлення сторінок системи, що тестується на предмет замаскованих полів в автоматичному режимі;
- Функція динамічного SSL (Secure Sockets Layer) надає юзерам сканеру ZAP змогу створювати персональний центр сертифікації, котрий здатний наказувати браузеру покладатися на нього. Це дозволяє сканеру легко отримувати доступ до трафіку, котрий захищено протоколом HTTPS;

- Після завершення аналізу певного ПЗ сканер ZAP здатний генерувати звіти, в яких буде відображено виявлені проблеми безпеки системи й можливі способи боротьби з ними.

Тестування якості й безпеки системи за допомогою сканеру ZAP можна розділити на умовні чотири етапи, котрі наведено у списку нижче [20]:

- Першим кроком для початку аналізу системи, що нас цікавить є визначення URL-адреси, за якою будуть здійснюватися атаки на систему;
- Другим кроком є процес проведення сканування й аналізу системи на предмет недоліків й вразливостей;
- Після завершення процесу сканування сканер ZAP виконує оцінку критичності виявлених вразливостей й проблем безпеки та розділяє знайдені вразливості на категорії за їх потенційною небезпекою для системи;
- Після проведеного етапу оцінки вразливостей та їх ризиків підбиваються підсумки й формується звіз з списком виявлених проблем та потенціальними їх вирішеннями.

2.2 Arachni

Arachni являє собою продукт для багатьох систем, котрий має високу працездатність, є легким у використанні рішенням для пошуку проблемних сторін веб-застосунків й здатний у повністю автоматичному режимі здійснювати пошук неприхованих джерел актуальних веб-застосунків. Створений на базі мови програмування Ruby. Основною ціллю під собою має допомогти людям, що тестують ПЗ й розробникам проєктів отримати визначити рівень захищеності систем. В Arachni присутній його особистий веб-браузер, котрий застосовуючи програмний модуль PhantomJS, має можливість виконувати перевірку складних веб-систем, що створені з використанням мови програмування JavaScript, HTML, CSS, AJAX (Asynchronous Javascript and XML) й DOM (Document Object Model)

[21]. Результат отримується завдяки перегляду переходів DOM, показників JavaScript й потоків здійснення.

В Arachni присутні наступні можливості:

- перевірка складових частин клієнтської частини веб-застосунків;
- робота з cookie;
- перегляд форм;
- перевірка параметрів запитів у форматах передачі даних XML (Extensible Markup Language) й JSON (JavaScript Object Notation);
- перегляд UI;
- перевірка даних запитів у форматах XML й JSON;
- перевірка вхідних даних UI;
- перевірка параметрів заголовків.

Контролювати процес виявлення вразливостей інструментом Arachni можливо завдяки терміналу й графічному інтерфейсу через веб-браузер.

На рисунку 2.2 представлено інтерфейс Arachni.

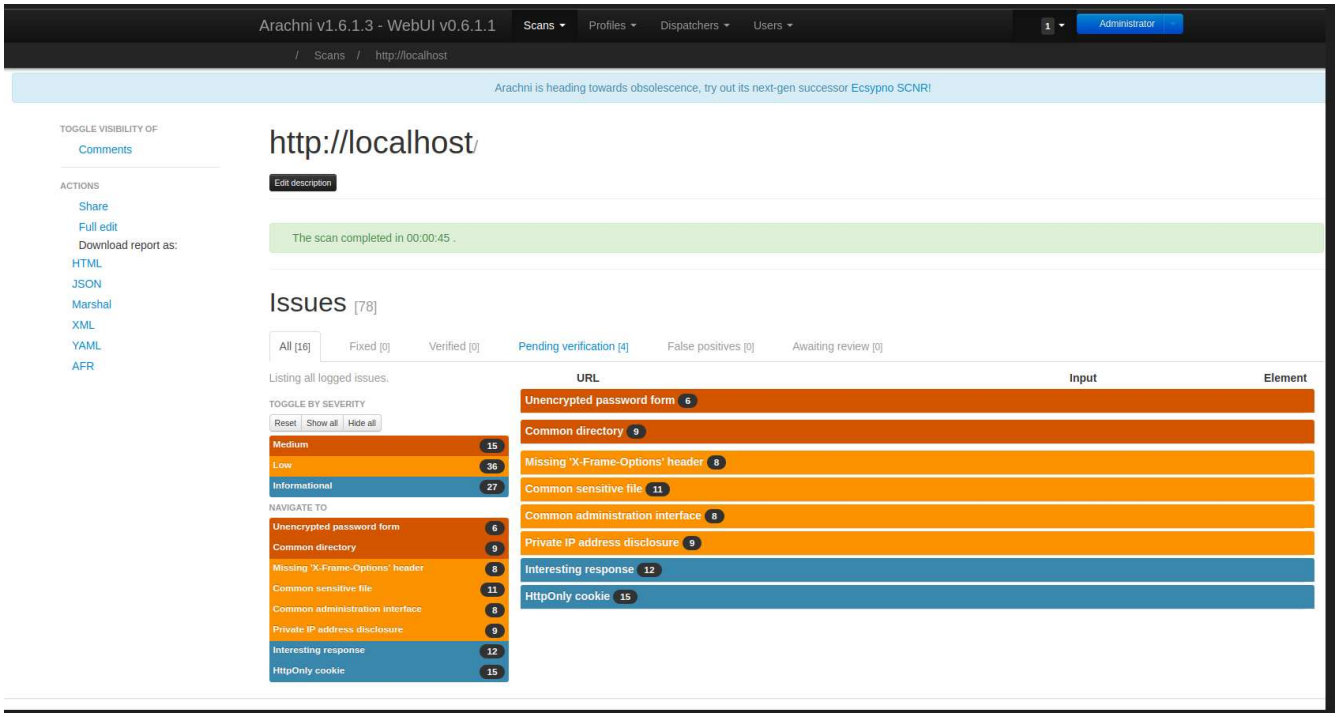

Рисунок 2.2 – Інтерфейс Arachni

Дана система має нескінченну кількість параметрів для налаштування, це стає прекрасним вибором для програмістів, котрі планують користування

динамічним скануванням. Серед параметрів можна виділити можливість гнучкої конфігурації особливостей використання форм, налаштування HTTP заголовків та cookie, можливість використання HTTP запитів в паралельному режимі. Також параметри, котрі котрі залишились, мають сумісність з SOCKS та HTTP-проксі й повноційну сумісність з SSL.

Сканування інструментом Arachni можливе лише завдяки методу тестування DAST. В системі Arachni присутня функція обробки та переліку користувацьких даних, але, нажаль, сенсу використовувати цю функцію досить мало. Також, в сканері відсутній пасивний вид проведення сканування, тому сканером можна робити тільки активне виявлення вразливостей.

Сканер Arachni використовує деякі методи аналізу систем, серед яких можна виділити використання аналізу багатопоточним методом, надсилання HTTP запитів в режимі асинхронності й паралельну їх обробку [22]. Arachni спроможний навчати сам себе, на основі отриманих відповідей, під час проведеного сканування. Його перевагою є можливість знаходити й встановлювати редагування, в процесі проходження через цикломатичну складність веб-систем, що здійснюється в зв’язку з звертанням уваги на мінливу сутність веб-застосунку. Внаслідок цього Arachni доводить свою перевагу над людським фактором та знаходить напрямки проведення атак з використанням вразливостей системи безпеки ПЗ.

Arachni має функцію створення звіту з результатами проведеної перевірки вразливостей системи. Звіти можна отримати у терміналі або у наступних форматах: XML, YAML (YAML Ain't Markup Language), HTML, JSON та TXT.

У звіті присутній перелік усіх проблемних сторін системи, що подається у вигляді графіків з наведенням визначеної інформації про кожен тип вразливості й можливі засоби їх профілактики. Різниця між активним та пасивним скануванням заключається в тому, що активне сканування зазвичай здійснює пошук CSRF (Cross-site request forgery) атак, атак ін’єкціями й вразливостей XSS.

Пасивне сканування, в свою чергу, аналізує некоректне налаштування системи захисту й проблеми забезпечення безпеки чутливих даних.

У Arachni закладено 65 типів сканувань, з яких 30 являються методами активного сканування й 35 способів аналізу системи в пасивному форматі. Поки активні методи аналізу відправляють запити й моніторить отримані від серверу відповіді, задля пошуку проблем безпеки, пасивні методі сканування слідкують за трафіком між сканером та ПЗ. Також, існують 19 додаткових модулів, котрі можуть надати змогу більш глибоко й детально просканувати ПЗ. Доступна можливість використання певних визначених конфігурацій тестування, використовуючи заздалегідь створені профілі для тестування систем на предмет вразливостей.

2.3 Wapiti

Програмне рішення Wapiti представляє собою інструмент виявлення порушень безпеки системи з відкритим кодом, котрий створено на мові програмування Python. Дане ПЗ можна використовувати тільки за допомогою інтерфейсу командного рядка, що завдяки саме такому методу роботи ПЗ робить його використання складною та неаргономічною задачею. Хоча сповіщення у терміналі можуть бути показані у різних кольорах, Wapiti за камфортом використання не може скласти концуренцію рішенням, що використовують графічний інтерфейс [23].

Wapiti це ще одне ПЗ, принципом роботи якого є метод чорної скриньки. Таким чином, інструмент виявляє вразливості й порушення безпеки систем без дослідження програмного коду програм, всього лише здійснюючи пошук серед усіх сторінок ПЗ на предмет вразливих сценаріїв й форм. Wapiti надсилає користувачу сповіщення, коли знаходить помилки в роботі застосунку, кожній з яких присвоюється власний код або номер. При проведенні аналізу веб-застосунку сканером Wapiti є можливість вносити чи відкидати дані під час пошуку вразливих форм або сценаріїв.

При користуванні даними засобами, Wapiti стає так називаємим фаззером та аналізує систему шляхом надсилання свідомо хибних даних або непередбачуваних даних, для того, щоб зруйнувати визначені сценарії або форми й перевірити їх на наявність відомих вразливостей [24]. Wapiti користується шляхом аналізу саме визначеної URL-адреси, що робить неможливим задання вхідних даних для створення запитів.

На рисунку 2.3 зображено програмний інтерфейс сканеру Wapiti.

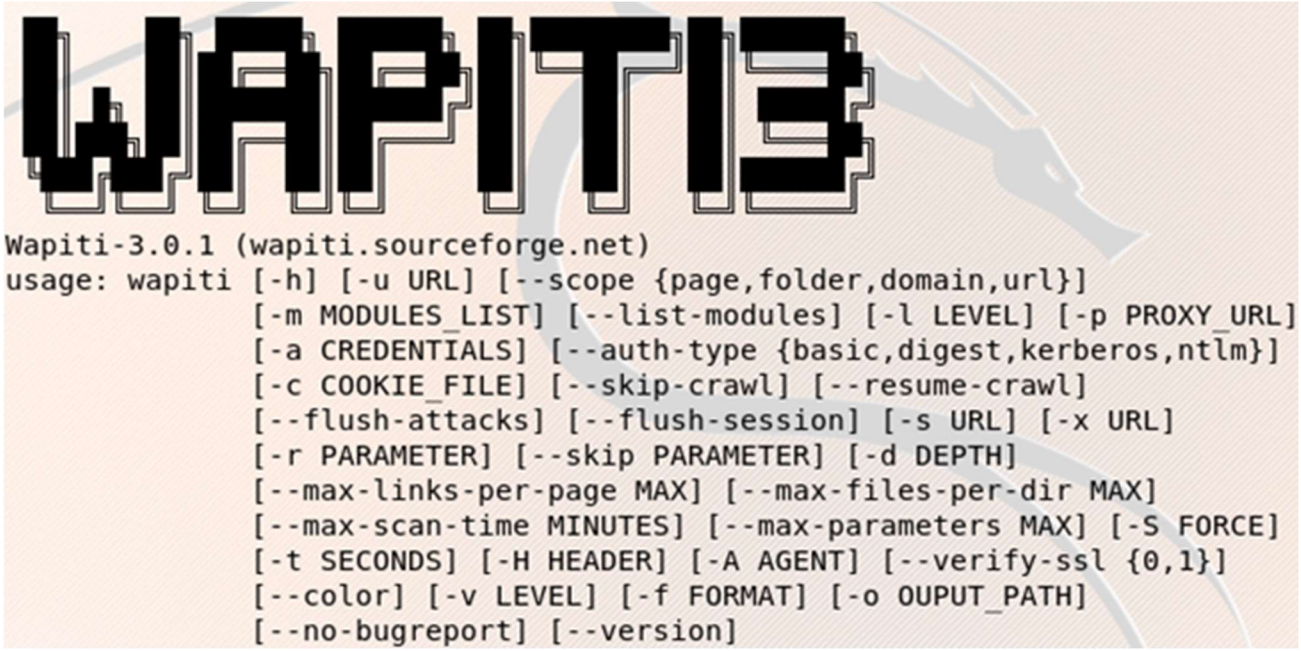

Рисунок 2.3 – Програмний інтерфейс сканеру Wapiti

Сканер надає різноманітні ступені багатослівності, завдяки чому присутня можливість легко включити та виключити модулі атак. Процес збільшення сприятливого навантаженя є таким же легким у реалізації, як написати рядок у тексовому полі. Wapiti може здійснювати сканування за HTTP методами POST та GET. Повноцінно працює з HTML5. Крім здійснення звичайного сканування інструмент має можливість проводити аналіз сумнівних серверних файлів, недоліків налаштування котрі можуть викликати вразливості й пошук небажаних файлів системи, котрі здатні власноруч зламати систему захисту веб-застосунків при тій умові, що хтось їх досягне. Може використовувати проксі видів HTTPS й HTTP.

Інструмент виявлення вразливостей Wapiti може виявити наступні вразливості механізмів безпеки систем:

- Атаки методом проведення SQL ін’єкцій, котрі направлені на БД;
- Можливість проведення атак на веб-додаток з використанням XSS ін’єкцій;
- Впровадження ін'єкцій CRLF;
- Перевірка на присутність конфіденційних або чутливих файлів;
- Можливість проведення XXE (XML External Entity) атак на систему;
- Виявлення проблем, що пов’язані з маніпулюванням файлами й можливою їх компроментацією;
- Виявлення виконання шкідливих команд;
- Можливість проведення атак методом підробки запитів юзера на боці серверної частини додатку, тобто SSRF атак;
- Перевірка системи на погані налаштування безпеки, через які можна впроваджувати вразливості;
- Можливість проведення атак з використанням LDAP ін’єкцій.

Серед форматів надання звітів про виконаний аналіз вразливостей можна виділити наступні: TXT, HTML, JSON та HTML.

Така варіативність надання звітів є гнучким та зручним рішенням для кінцевих користувачів інструменту. Увесь звіт є максимально коротким та чітким. Відсутня можливість продивитися повний звіт сканування з усіма тонкощами результатів скану вразливостей. Звіт, що представлений у форматі HTML надає дані про проведене виявлення вразливостей у таком ж виді, який присутній у переважній кількості попередніх інструментів, детально показуючи деталі надісланих запитів й отриманих на них відповідей, виявлені типи вразливостей й недоліків систем безпеки та особливо вразливих боках системи.

2.4 Nikto

Ціна інструменту сканування Nikto на ринку інструментів виявлення вразливостей дорівнює нулю. Даний інструмент виявлення вразливостей веб-

застосунків та недоліків мережевої безпеки виконує роботу в автоматичному режимі. Функціонал й методики сканування регулярно оновлюються, в тому числі в автоматичному режиму, що дозволяє користувачам отримувати доступ до кращих практик боротьби з недоліками систем безпеки веб-додатків. Присутня можливість проведення загального сканування серверів для більшості складових частин.

В базі сканеру заздалегіть приготовлені більш ніж 60000 сумнівних файлів, котрі містять параметри небезпечного файлу, функції пошуку оновленних версій серверів. Також є можливість виконати сканування параметрів налаштування серверу, серед яких є присутність якого-небудь числа індексних файлів та параметрів серверної частини програми, що працює за HTTP запитами.

Сканер виконує свої задачі на базі платформи Linux, основа системи реалізована на мові програмування Pearl. Може виконувати сканування веб-застосунків тільки при керуванні з терміналу, не маючи доступу до керування через графічний інтерфейс. Попри це, для створення й початку сканування терміналу вистачає сповна. Виконує перевірку конфігурації веб-сервер, серед яких можна виділити перевірку на присутність деякої кількості індексних файлів, налаштування HTTP веб-серверу й встановлення застосунків [25]. Присутнє автоматичне оновлення системи.

На рисунку 2.4 представлено програмний інтерфейс сканеру Nikto.

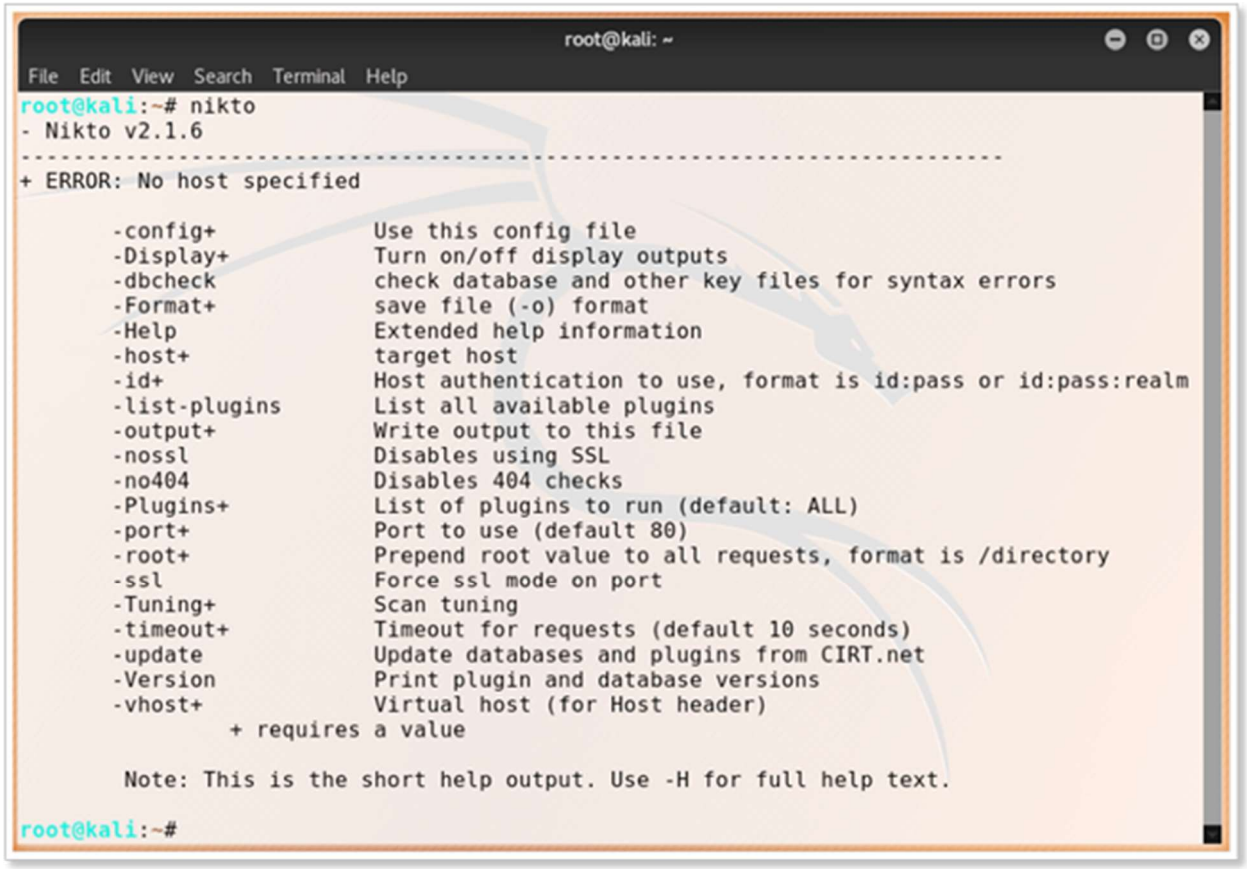

Рисунок 2.4 – Програмний інтерфейс сканеру Nikto

Завдяки застосуванню інструменту присутня можливість визначити напрямки атак, серед яких є як XSS атаки, так і використання різних дій з файлами, які можно використати, щоб покласти сервер. У Nikto пристутні різномантні алгоритми налаштування модулів. Також наявні численні модулі для покращення пошуку проблемних місць у системі завдяки визначеним підходам відповідно до типів вразливостей. Nikto позиціонує себе як не перевантижений та прости у використанні сканер. Дана утиліта робить аналіз файлів налаштування серверів, а саме індексних файлів й конфігурує виклики HTTP. Це чудовий засіб для початку роботи зі сканером завдяки використанню SSH віддалено.

Nikto в процесі тестування системи через проведення запитів на сервер й отримання зворотного зв’язку проводить відповідний аналіз відповідей отриманих з сервера. У сканері Nikto присутня функція надсилання інформації через запити, що робить можливим знаходження проблемних сторін у системі,

серед яких є SQL-ін’єкції та XSS [26]. Серед додаткових параметрів роботи програми можна виділити можливість виконувати аналіз завдяки проксі та SSL протоколу. Звіти про виконане сканування з результатами виявлених вразливостей надаються у досить скромній кількості форматів, серед яких є: CSV, HTML й XML. Завдяки використанню управління через термінал, інструмент можна зручно оновлювати, забезпечуючи кращий захист систем.

2.5 Burp Suite

Burp Suite представляє з себе ПЗ, що симулює атаки на веб-застосунки. Сканер створенний на мові програмування Java, через це він кросплатформлений для середовищ Java. ПЗ належить компанії «PortSwigger». Для користувача існує дві версії: Повністю безкоштовна та платна з повним доступом до всіх функцій, але з можливістю використання пробної версії. ПЗ доступна на таких ОС як: Windows, Linux й MAC. Для таких людей, котрі виконують аудит системи безпеки веб-додатків й тестувальників ПЗ є незамінним рішенням у вирішенні більшості критичних проблем безпеки.

Якщо говорити про локальні веб-проксі, то це Burp, його можливості надають спроможності для пошуку, перевірки та який дозволяє перехоплювати, контролювати та міняти HTTP та HTTPS запити, а також те ще отримує браузер користувача від веб-застосунку [27]. ПЗ надає інформацію про можливі напрямки атаки на додаток.

У Burp є інструмени, котрі надають юзерам можливість створювати його особисті модулі. Також, пристунє пасивне й активне види сканування вразливостей. Інтерфейс Burp є ергономічним та інтуїтивно зрозумілим, тому користування програмою не викликає ніяких проблем навіть у недосвідчених користувачів. В той же час сканер має велику кількість критично важливих й ефективних функцій для виявлення вразливостей, котрі дозволяють професіоналам виконувати їх роботу максимально ефективно та точно.

Після проведення тестування системи інструмент розділяє виявлені загрози на категорії й призначає оцінку достовірності кажній з них. Через це Burp

можливо назвати кращим сканером для надання інформації щодо стану системи безпеки й оцінки проблемних сторін веб-застосунка.

Головною метою роботи сканеру Burp є перекриття й подальший показ повідомлень, що були передані за протоколом HTTP в оргаізованому виді, що дозволяє користувачам інструменту тестування одержати звіт по всій системі з мінімальними затримкаму у часі. У сканері Burp присутній UI, що надає змогу здійснювати повний контроль за інформацією про систему та можливість маніпулювання такою інформацією. В такому разі, функціонал сканеру можна описати як побудова юзером планів виконання атак на веб-застосунок. Далі юзер вивчає й оцінює отримані результати. Йде аналіз запитів з серверу ПЗ на існування потенційних загроз та проблем безпеки.

Сканер усвідомлений про всі користувацькі дії, маніпуляції та переміщення, що дозволяє наглядно бачити комунікації між клієнтом й сервером. Головним плюсом сканеру Burp над іншими сканерами є те, що всі корисні й ефективні інструменти забезпечення безпеки веб-систем зібрані в купу в особі однієї програми. Присутня можливість оцінки й корегування POST й GET запитів, задля контролю ПЗ. Позиціонує себе як інструмент з використанням проксі, котрий виконує індексацію URL-посилань, що було переглянуто, після цього йде виставлення їх до черги, щоб провести процес оцінки вразливостекй. Сканеру треба виконувати алгоритми й сценарії у системі. Прикладом таких виконань може бути авторизація, введення даних й рух по веб-сторінках.

На рисунку 2.5 представлено програмний інтерфейс сканеру BurpSuite.

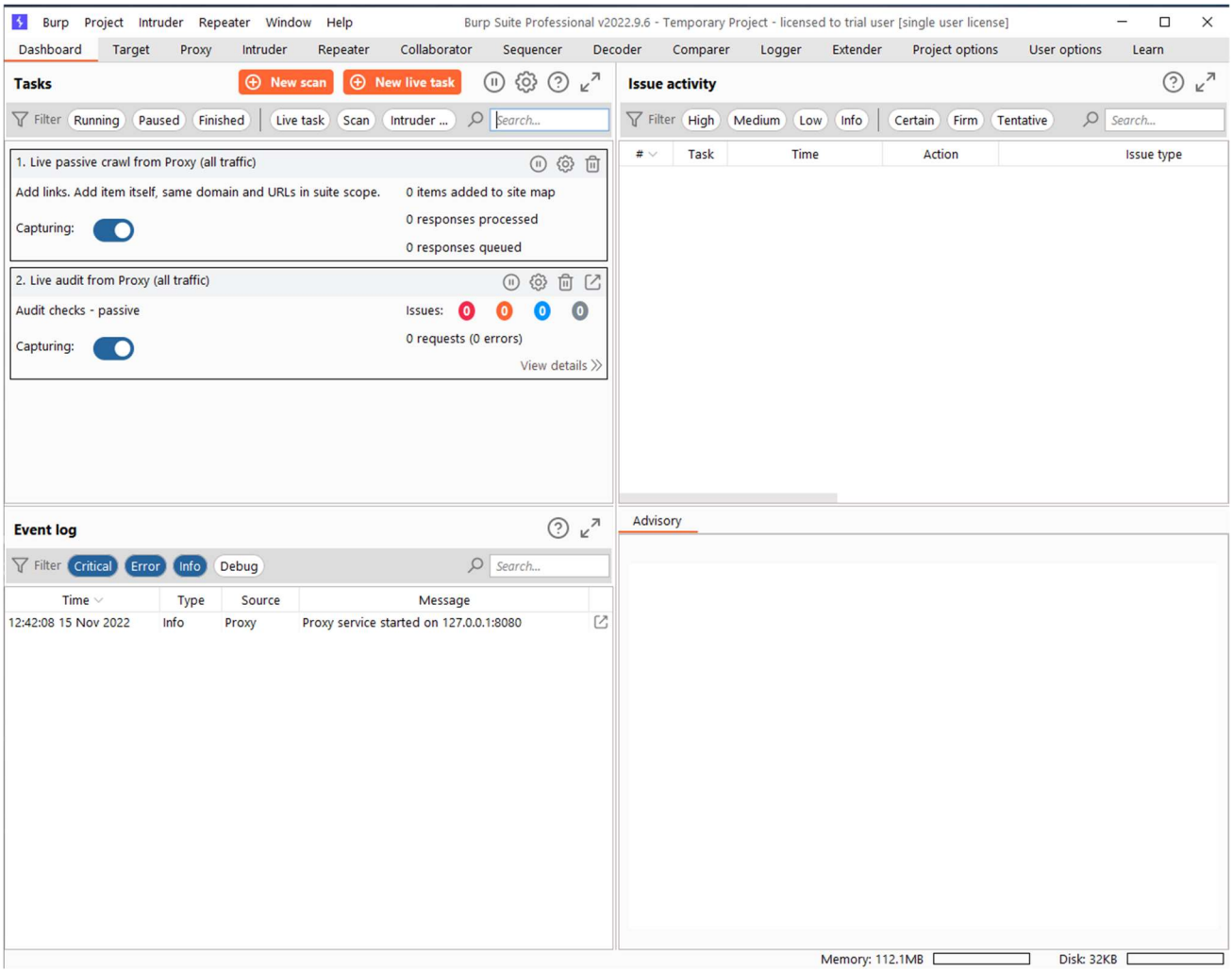

Рисунок 2.5 – Програмний інтерфейс сканеру Burp

Такі корисні особливості як деводування й робота з повідомленнями, на жаль, доступні тільки для запуску в автоматичному режимі роботи інструменту Burp. В мануальному режимі роботи використовувати можна лише базовий нескладний функціонал, як, наприклад, маніпулюції з параметрами. Мануальна оцінка всіх HTTP-повідомлень з точки зору затраченого часу не є вигідним рішенням.

Для реалізації більш глибоких планів й сценаріїв на проведення атак сканер дозволяє використовувати певні розширення, котрі допомагають реалізовувати додатковий функціонал для таких атак [27]. Алгоритми Burp спроможні запровадити спостереження і оцінку повідомлень HTTP на будь-який смак. Алгоритми спроможні, як і створювати нові складові частини UI, так і змінювати існуючі для їх подальшого показу. Burp складається з декількох

важливих функцій, котрі надають можливість виконувати роботу в умовах аналізу сучасних складних веб-систем. Завдяки використанню гнучкого власного API й варіанту розширення свого функціоналу, Burp надає не тільки великі перспективи для ручного тестування, а й потужні алгоритми й можливості для проведення автоматичного сканування.

Далі представлено головний функціонал інструменту Burp Suite [28]:

- Функція Target являє собою таку функцію сканеру Burp, котра спроможна при необхідності зібрати всі ресурси веб-застосунків, таким чином ведучи за руку користувача протягом всього процесу аналізу системи.
- Декодер надає можливість маніпулювати кодуванням даних. Дана функція працює завдяки можливості реалізації різних методів кодування та використання типових хеш-функцій.
- Використання проксі надає юзеру можливість отримувати доступ до трафіку, котрий проходить як через сервер так і через UI й контролювати необроблений трафік й здійснювати необхідні маніпуляції з ним. Надає представлення трафіку у форматі «сирого тексту» або таблиці з параметрами. Трафік котрий йде з обох сторін при необхідності ефективно проціджується завдяки алгоритмам фільтрування з використанням можливості об'єднання алгоритмів через нескладні набори інструкцій. Історія трафіку сберігається в системі, де з нею можливі подальші маніпулювання. Присутня можливість роботи з заголовками, що дає змогу конфігурувати частину рядків в HTTP заголовках.
- Функція сканування створена для сканування можливих загроз веб-застосунків в автоматичному режимі, але її можливо використовувати тільки у платній версії Burp.
- Функція Intruder надає можливість здійснювати корегування й забезпечувати надсилання запитів в автоматичному режимі. Процес

фазингу реалізується завдяки здійсненню аналогічного запиту декілька раз з використанням різних даних, котрі мають бути непередбачені для тестованої системи, що дозволяє ефективно шукати проблемні сторони механізмів захисту ПЗ.

- Функція Sequencer - це система аналіку маркерів безпеки та файлів cookie застосунків.
- Функція Repeater являє собою інструмент для мануального проведення запитів за HTTP й HTTPS протоколами, з додаванням деякого їх налаштування. Перед відправкою кожного нового запиту до нього можна дуже швидко внести необхідні для тестування зміни. Завдяки системі логування запитів, можна легко переглядати інформацію, що пов’язана й отримані на них відгуки, й далі можна гнучко шукати необхідні значення. Іншими словами, інтегрована система пошуку надає можливість спостерігати зміну значень, через поставлені відповіді.
- Функція Spider дозволяє наглядно відобразити й перелічити всі можливості сканованого ПЗ. Її робота представляє з себе процес пошуку гіперпосилань в JavaScript й HTML формах та їх подальше застосування, з використанням допомоги у вигляді переліків каталогів й файлів налаштувань. На момент закінчення роботи фінальний результат надається або у вигляді таблиці й в той же час запити є можливість знову переправити до зломщика. При виборі однієї складової частини, система покаже звіт, де можна буде побачитии повідомлення про адресу посилання на ресурс та наявність решти гіперпосиланнь.
- Функція порівняння дає зрозуміти, які дані оновилися між веб-сторінками, що можна використовувати для наглядного відображення здійсненних змін.

3 ЗАСТОСУВАННЯ СКАНЕРІВ ВЕБ-ДОДАТКІВ

В ході виконання аналізу веб-додатку було обрано інструменти сканування, котрі були розглянуті раніше, а саме: Nikto2, Wapiti, Arachni, ZAP й Burp. Всі сканери працюють за методом тестування DAST, який часто називають методом чорної скриньки. Плюсом цього являється те, що даний підхід до тестування на наявність вразливостей дозволяє забезпечити один и той самий сценарій для проведення атак на веб-додаток, що в той же час не потребує знань про те, як влаштований додаток зсередини.

Оцінювалися результати за виявленими вразливостями згідно актуального списку найкритичніших вразливостей та загроз OWASP Top Ten 2021 року. Результати та їх точність були оцінені за відповідними метриками, котрі будуть розглянуті у цьому розділі нижче.

3.1 Сканування вразливого клієнт-серверного веб-додатку

Для проведення сканування веб-додатку відповідними інструментами на предмет вразливостей вони були запущені проти спеціальної програми, котра спеціально нашпигована вразливостями, з метою оцінки можливостей інструментів сканування. Ця програма називається DVWA (Damn Vulnerable Web Application) та розшифровується як «До біса вразливий веб-додаток». Реалізована програма на базі мови програмування PHP та MySQL.

DVWA є вразливим до наступних вразливостей [29]:

- атаки «грубою силою»;
- неефективні індикатори сеансів;
- атаки завдяки застосуванню ін’єкцій команд;
- CSRF атаки;
- завантаження файлів;
- SQL ін’єкції;

- XSS вразливості, котрі відображаються, зберігаються й пов’язані з DOM;
- незахищена Captcha;
- уникання політики безпеки вмісту;
- інклюзія файлів тощо.

Метою тестування DVWA на предмет вразливостей є подальше порівняння інструментів тестування для виявленні найкращих з них.

3.1.1 Застосування інструменту Wapiti

Зважаючи на роботу сканеру тільки в режимі командного рядку, сканер було незручно використовувати. Такий користувацький інтерфейс потребує відповідних навичок роботи з терміналом та деякого терпіння для вивчення його можливостей та отримання кращих результатів роботи сканеру. Дещо спростило цю прикрість та покращила роботу зі сканером можливість зміни кольору повідомлень. Wapiti працює тільки за DAST методом тестування та не має змоги працювати з вихідним кодом DVWA, що стало суттєвим її недоліком.

Задля того, щоб робота сканера зайняла меншу кількість часу, нами було використано тільки потрібні нам, конкретно у нашому випадку, модулі сканеру. За підсумком, сканер виявив вразливості та розділив їх на категорії 3 категорії.

На рисунку 3.1 зображено результат одного зі сканувань вразливого веб-додатку інструментом Wapiti за допомогою терміналу.

Рисунок 3.1 – Результат однієї зі спроб сканування інструментом Wapiti

Щоб працювати з автентифікацією в автоматичному режимі Wapiti має можливості виробляти cookie згідно з URL-адресою сторінки отримання доступу до застосунку. Через те, що cookie містять у собі деяку інформацію, інструмент сканування має можливість використовувати цю інформацію для процесу автентифікації. Під час кількох випадків виконання виявлення вразливостей у додатку DVWA, Wapiti не зміг оминути сторінку автентифікації. Задля оминання сторінок виходу та припинення процесу виявлення вразливостей, можна було використати загальне виключення деяких URL-адрес з роботи сканеру.

В ході роботи було виявлено, що сканер має малий час виконання процесу виявлення загроз, через те що пропускає якусь кількість параметрів та має мало заздалегідь прописаних корисних навантажень, котрі сильно псують можливості сканування даним сканером. Після завершення виконання тестування додатку на вразливості було отримано звіт з результатами роботи у форматі HTML.

3.1.2 Застосування інструменту ZAP

Попри те, що сканер від OWASP - ZAP надає можливість використовувати чотири режими роботи, а саме: безпечний режим, захищений режим, стандартний режим й режим атаки, тестування виконувалося в стандартному режимі, котрий передбачає тестування з використанням методу GET. Для відображення критичності виявлених загроз інструмент сканування ZAP показує біля кожної виявленої загрози відповідний їй кольоровий прапор, серед яких червоний відповідає найбільш критичним загрозам безпеки, помаранчевий - загрозам середньої критичності, жовтий – загрозам незначної тяжкості, а синій колір в свою чергу інформаційні застереження про слабкі недоліки веб-додатку.

В ходу сканування інструментом ZAP було виявлено проблем незначної критичності, середньої критичності й вразливостей високої критичності й розділено їх на категорії.

На рисунку 3.2 представлено результат одного зі сканувань вразливого веб-додатку інструментом сканування OWASP ZAP.

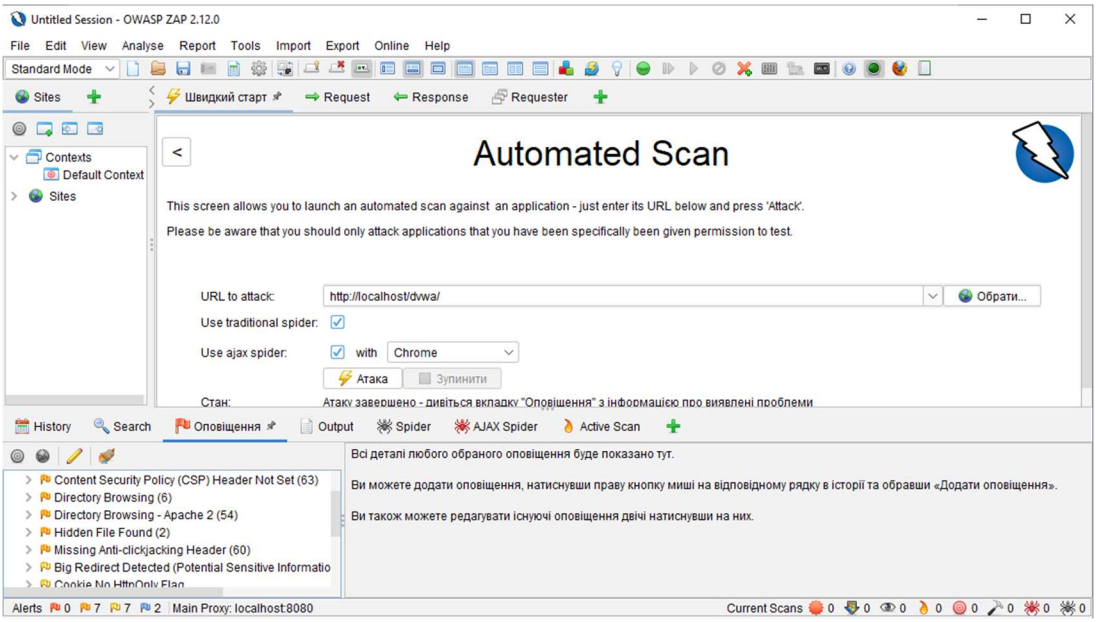

Рисунок 3.2 – Результат однієї зі спроб сканування інструментом ZAP

Виявлені випадки найбільшої критичності серйозності включають в себе атаки SQL-ін’єкціями, можливість атак за міжсайтовим сценарієм XSS, ризик міжсайтової підробки запитів CSRF, проблеми з HTTP заголовками та інші проблеми.

В цей же час число повідомлень про критичність не треба напряму асоціювати з числом проблем, котрі було знайдено в категоріях празливостей через те, що кожна категорія має можливість мати деяке число вразливостей проблем, але всі такі вразливості можуть знаходитися під одним прапором.

3.1.3 Застосування інструменту Nikto

Сканер виявлення вразливостей веб-систем Nikto крім вже звичних нам категорій критичних вразливостей також здійснював пошук деякої критично важливих даних про безпеку веб-додатку. Серед них можна виділити шифрування даних разом з протоколом SSL, що зазвичай використовується в мережі для забезпечення відповідного рівня безпеки й системну серверну інформацію.

А після завершення виконання сканування й отримання результатів було виявлено, що можливості інструменту виявлення вразливостей Nikto є дещо кращими за його прямого конкурента ZAP, що в свою чергу робить сканер кращим рішенням підтримки гарного рівня безпеки веб-додатків.

На рисунку 3.3 представлено Результат однієї зі спроб сканування інструментом Nikto.

Рисунок 3.3 – Результат однієї зі спроб сканування інструментом Nikto

Вже після виконання коду Nikto показав нам велику кількість недоліків безпеки системи, пов’язаних із заголовками. Відповідні проблеми можуть відповідати за можливість атак ін’єкціями, способом клікджекінгу, використання міжсайтового скриптингу та інші.

В ходу сканування було виявлено недоліки системи й розділено їх на 4 категорії за їх критичністю. Серед виявлених проблем можна звернути увагу на міжсайтовий скриптинг, тобто XSS, можливість маніпулювання діями користувача, тобто клікджекінг та інші проблеми.

3.1.4 Застосування інструменту Arachni

Для того, щоб сканування обраного вразливого веб-додатку сканером Arachni пройшло вдало, для проведення атак було задано конкретні URL-адреси й вказано перевірки, котрі мають бути виконані. Для керування інструментом аналізу було застосовано інтерфейс терміналу та використано відповідні модулі й форма динамічного аналізу додатку, для оптимізації надсилання даних. Також було задано не виконувати ті перевірки, котрі не відносяться до тих категорій, що нас цікавлять.

По завершенню перевірки DVWA на предмет вразливостей, сканер розприділив результати за їх критичністю, де червоний колір відповідає високій критичності виявленої вразливості, помаранчевий колір – середньому рівню критичності, жовтий - малому рівню критичності й, останній, синій – інформаційній категорії вразливостей. Також, для того щоб показати проблемні елементи HTML сторінки, котрі не являються якимись значними проблемами безпеки про які потрібно було б повідомити, використано було саме зелений.

На рисунку 3.4 зображено результат однієї зі спроб сканування інструментом Arachni.

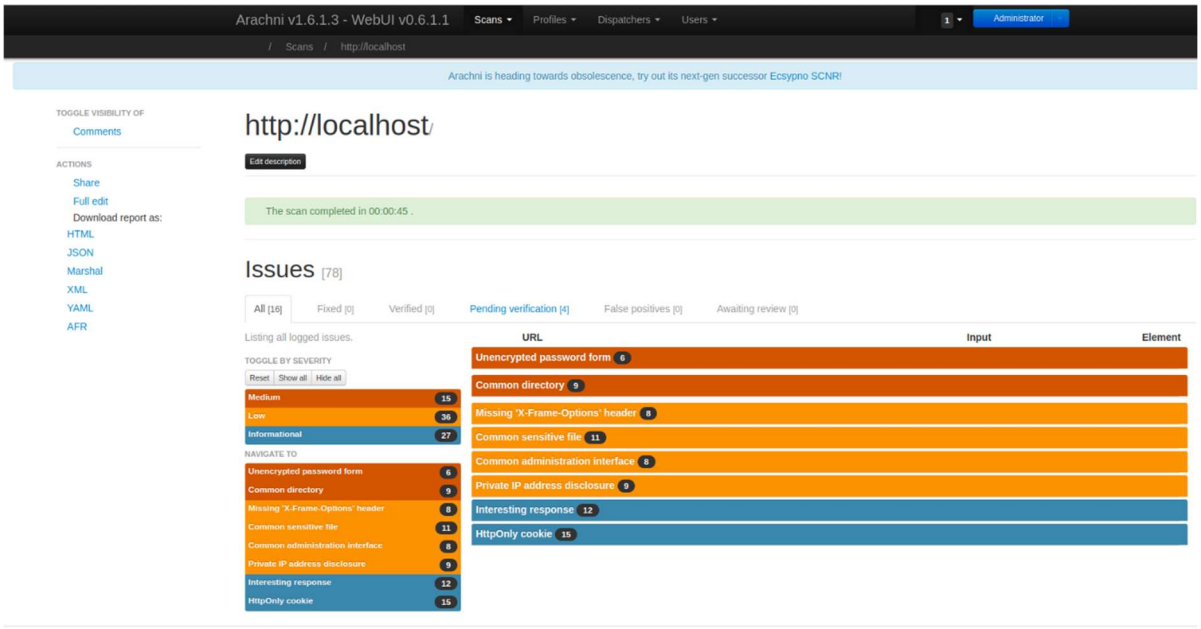

Рисунок 3.4 – Результат однієї зі спроб сканування інструментом Arachni

Чим більше запитів надсилається сканером, тим більш точним буде результат сканування й, зазвичай, тим більшу кількість проблем безпеки буде виявлено. Саме тому, ця особливість напряму позначається на зручності роботі з інструментом тестування. Оскількі результати процесу виявлення вразливостей дежо різнилися після кождого з ним, то було зроблено відповідний висновок, що сканер не є абсолютно надійним інструментом та інколи може упускати й не повідомляти про деякі важливі нюанси безпеки. Також, у офіційній інструкції до Arachni було вказано про деяку автоматичну адаптивність сканеру до певних веб-програм, що могло вказатися на проблемі з відрізняючимися результатами роботи сканеру від сканування до сканування.

3.1.5 Застосування інструменту Burp Suite

Сканер BurpSuite був використаний нами для сканування веб-додатку DVWA. Для цієї задачі Burp дозволяє виконувати динамічний аналіз на вразливості, яким ми й скористалися. Першим кроком ми додали цікавлячі нас URL-адреси до відповідного поля. Це потрібно для того, щоб сканер почав процес сканування загальної цільової системи, на сканування якой ми і націлені. Сканування працює за методом роботи з кожною можливою сторінкою веб-програми та атак різними методами, серед яких велика увага приділяється SQL

ін’єкціям, для того щоб побачити які саме місця програми є найбільш вразливими.

На рисунку 3.5 зображено результат однієї зі спроб сканування інструментом Burp.

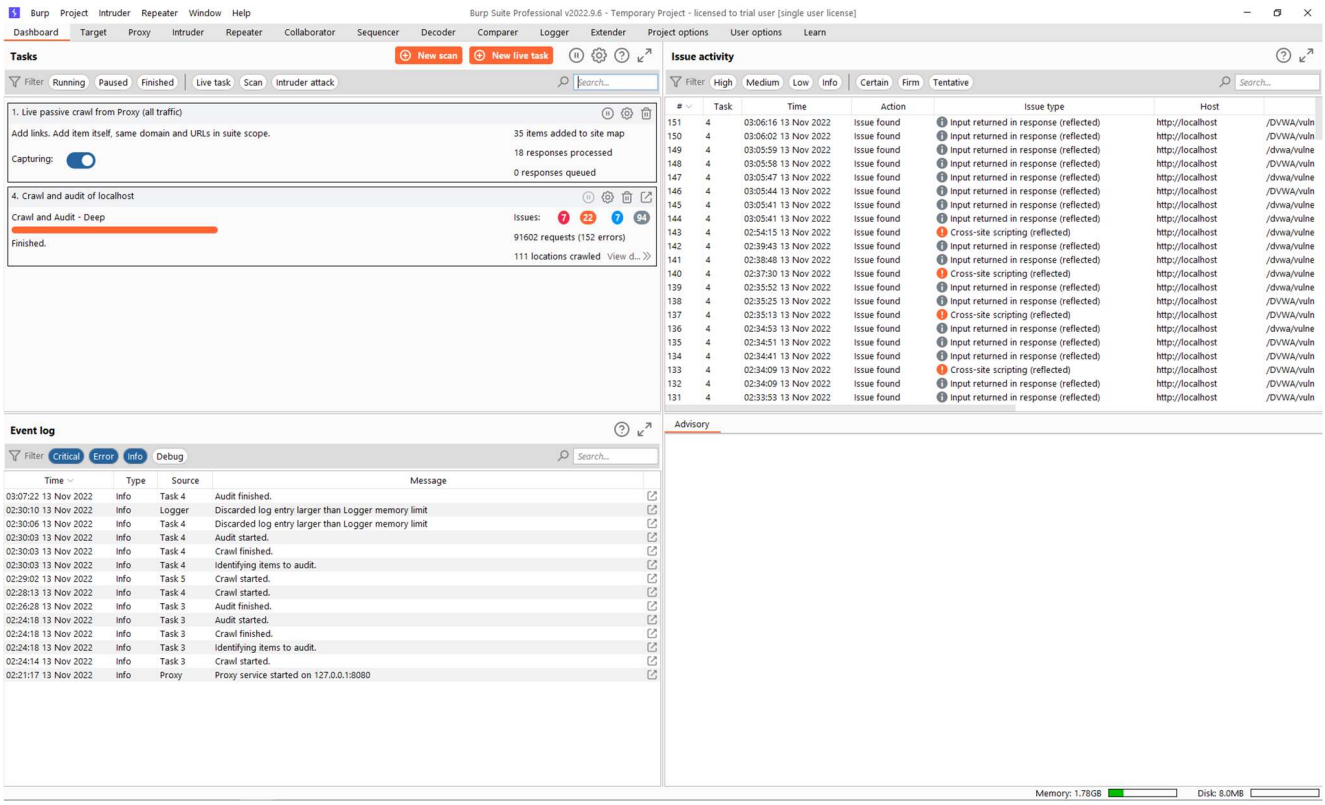

Рисунок 3.5 – Результат однієї зі спроб сканування інструментом Burp Suite

В процесі виконання своєї роботи Burp показував різні оповіщення про виявлені помилки. Загальний процес сканування потребував багато часу, через ретельний пошук вразливостей, перебирання усіх сторінок програми, тому у результаті роботи можна бачити 96000 відправлених запитів. Кожна виявлена проблема системи далі класифікується на основі проблеми та її усунення. Загалом завдяки Burp було виявлено й класифіковано понад 100 вразливостей, які було розділено на чотири категорії критичності. Пропрацював інструмент приблизно 50 хвилин.

Було проаналізовано трафік, щоб виявити вектори атак. Аналіз пакетів підтвердив, що більшість часу сканери вводять однакові дані для полів пароля та повторного пароля під час створення користувачів. Було виявлено як XSS у

вигляді підробку міжсайтового запиту та впровадження заголовків, так і SQL-ін’єкції, а саме: тавтології, неправильні запити та запити на об’єднання та обхід каталогу. Було також помічено можливу ін’єкцію XML, яка є хибно-позитивною. Знайдено вразливості незашифрованого з’єднання та надсилання пароля у відкритому вигляді. Burp визначив діапазони IP-адрес із незашифрованим зв’язком, деякі з яких дозволяють передавати паролі через незашифровані мережі.

Burp повідомляє про меншу кількість хибних спрацьовувань порівняно з іншими інструментами. Ефективність часу, витраченого на всі ці інструменти, дуже залежить від надійності їх висновків.

3.2 Аналіз отриманих результатів виявлення вразливостей

В даному розділі можна бачити результати, котрі було отримано зважаючи на проведені аналізи вразливого веб-додатку різними сканерами виявлення веб-вразливостей. Сканери було оцінені за наступники показниками роботи: кількість виявлених вразливостей за відповідними категоріями зі списку OWASP, ступінь критичності виявлених вразливостей та точність отриманих результатів.

Оцінка інструменту виявлення наявних вразливостей веб-додатків як таких напряму залежить від правильності виявлених результатів та точності проведеного тестування. Серед метрик характеристик якості роботи сканерів можна виділити наступні: правильність сканування на предмет вразливостей, якість проведеного фаззингу й якість аналізу веб-додатку.

3.2.1 Порівняння результатів роботи інструментів сканування додатку

В ході проведення сканування веб-додатку на предмет наявних у ньому вразливостей різними інструментами виявлення вразливостей було отримано деякі результати.

Сканування являє собою деякий процес визначення вразливостей веб-додатку на кожній з існуючих сторінок застосунку. Чим більше сторінок перевірить сканер, тим кращим та точним буде проведене сканування.

Оцінювання ефективності роботи інструментів сканування веб-додатку відбувалося за принципами виявлення вразливостей за категоріями найпопулярніших критичних вразливостей OWASP Top Ten.

Порівняння результатів роботи сканерів можна бачити у таблиці 3.1.

Таблиця 3.1 – Порівняння сканерів вразливостей веб-додатків за результатами пошуку вразливостей

Назва критерію оцінювання	*Назва інструменту сканування*				
	ZAP	*Arachni*	*Wapiti*	*Nikto*	*BurpSuite*
Порушений контроль доступу	7	8	2	5	8
Криптографічні збої	0	0	0	0	0
Ін’єкції	24	24	22	21	26
Небезпечний дизайн	3	1	1	0	1
Неправильна конфігурація безпеки	2	5	5	1	4
Вразливі та застарілі компоненти	0	8	1	3	8
Помилки ідентифікації та автентифікації	47	51	64	38	51

Продовження таблиці 3.1

Порушення цілісності даних та ПЗ в цілому	4	2	3	2	16
Помилки в моніторингу на логуванні проблем безпеки	1	4	0	2	2
Підробка запитів на стороні сервера	10	7	1	12	7

3.2.2 Точність проведених результатів

Фаззинг являє собою дію, котрі передбачають собою введення даних. Якість такого процессу має пряму залежність від початкових даних, котрі користують недоліки системи задля проведення тестування та в той же час не визначаються сервером. Іншими словами, фаззер входить використовуючи відповідні вхідні дані, щоб перевірити систему на вразливості.

У свою чергу за результати після процессу фаззингу та оцінки таких результатів відповідальним є аналізатор. Він безпомилково аналізує та визначає виявлені вразливості.

3.2.3 Метрики оцінки результатів сканування додатку

Інструменти виявлення вразливостей, що працюють за методом DAST здатні й часто помиляються, показуючи некоректні результати, котрі ми й називаємо хибно-позитивними. Так відбувається, тому що такі інструменти приділяють велику увагу перевірці на відповідність підписів. Таким чином інколи й відбуваються невдачі з точки зору знаходженні вразливостей. Опираючись на цей факт, користувачам інструментів сканування методом чорної

скриньки вкрай необхідно, задля отримання безпомилкових результатів, перевіряти отримані результати самостійно. Для перевірки отриманих в результаті сканування результатів можна використовувати методи, котрі передбачають використання таких концепцій, як: хибно-негативні (ХН), тобто ті, що не були виявлені інструментом сканування взагалі й хибно-позитивні (ХП), що в свою чергу, були знайдені й показані помилково, вразливості.

Такий параметр інструменту виявлення вразливостей як Precision, який можна вважати точністю результатів проведеного сканування, можна представити у вигляді формули й порахувати. Таким чином, точність буде розраховуватися так, що відсоток істинно вірних результатів, тобто СП, потрібно поділити на суму загального числа знайдених недоліків безпеки системи СП й хибних виявлень ХП. Відповідна формула представлена нижче.

$$\text{Точність} = \frac{\text{СП}}{\text{СП} + \text{ХП}}$$

У таблиці 3.2 представлені результати проведеного тестування з урахуванням таких показників, як справжні-позитивні (СП) результати, хибно-позитивні результати (ХП) й розрахована точність проведеного інструментами виявлення вразливостей сканування.

Таблиця 3.2 – Точність проведеного сканування вразливостей веб-додатків

Інструмент сканування	*XSS*		*SQLI*		*Показник точності*
	СП	*ХП*	*СП*	*ХП*	
BurpSuite	17	1	8	0	96%
Nikto	11	2	6	2	81%
Arachni	17	2	5	0	92%
Wapiti	13	3	5	1	82%
OWASP-ZAP	17	0	7	0	100%

Показник якості виконаного виявлення вразливостей Recall являє собою деякий відсоток відкликань проведеного тестування або, іншими словами, чутливість. Його можна представити у вигляді формули й порахувати. Щоб

правильно розрахувати даний показник, нам потрібно знати лише кількість справжніх-позитивних виявлень вразливостей, тобто СП й хибно-негативних виявлень, тобто ХН. Нажче представлено відповідну формулу розрахунку показника чутливості, тобто Recall.

$$\text{Чутливість} = \frac{\text{СП}}{\text{СП} + \text{ХН}}$$

У таблиці 3.3 показано наглядний розрахунок параметру чутливості, згідно з проведеним раніше скануванням веб-додатку різними на предмет вразливостей XSS та SQLI. Наглядно можна бачити відсоток відкликань, що було розраховану з урахуванням справжніх виявлень та хибних виявлень вразливостей.

Таблиця 3.3 – Порівняння параметру чутливості інструментів сканування вразливостей веб-додатків

Інструмент сканування	*XSS*		*SQLI*		*Показник чутливості*
	СП	*ХН*	*СП*	*ХН*	
BurpSuite	17	0	8	0	100%
Nikto	11	6	6	2	68%
Arachni	17	0	5	3	88%
Wapiti	13	4	5	3	72%
OWASP-ZAP	17	0	7	1	96%

4 ПРАКТИЧНІ РЕКОМЕНДАЦІЇ

Опираючись на розгляд інструментів виявлення вразливостей веб-додатків у 2 розділі й тестування вразливого веб-додатку з аналізом отриманих результатів у 3 розділі виробимо практичні рекомендації щодо доцільності використання інструментів сканування в цілому й кожного з розглянутих та використаних сканерів зокрема.

На підставі проведеного аналізу вразливого веб-додатку інструментом виявлення вразливостей Wapiti виявив найбільш кількість вразливостей, пов’язаних з ідентифікацією й автентифікацією, загальна точність виявлення вразливостей в ході проведення XSS та SQL атак склала усього 82 відсотки, що є одним із найгірших результатів за цим показником. В результаті аналізу результатів використання Wapiti було виявлено, що трохи більше 18 відсотків виявлених вразливостей є хибними, що не грає йому на руку. Не виявленими залишились цілих 28 відсотків вразливостей категорії вразливостей Injection за рейтингом OWASP 2021 року, що стало відносно поганим результатів, гіршим за який став тільки результат роботи сканеру Nikto. Комфорт використання сканеру не став вагомим недоліком інструменту, але може стати для інших користувачів, котрі не знайомі з терміналом. Аналіз веб-додатку було завершено за малу кількість часу, зважаючи на те, що сканер має низьке число заздалегідь визначених корисних навантажень й здатний пропускає деякі параметри сканування. Таким чином, Wapiti можна вважати відносно неактуальним в порівняння з іншими його конкурентами.

В результаті роботи сканеру Nikto було отримано невтішні результати. Сканер став гіршим за своїх конкурентів майже у всіх категоріях виявлених загроз, обійшовши їх тільки за виявленням SSRF вразливостей. В прямому ж порівнянні інструментів за знайденими XSS та SQL загрозами, показник точності виявлених вразливостей склав 81 відсоток, що стало найгіршим результатом серед усіх сканерів. Це говорить про велику кількість хибно-позитивних

виявлень загроз. Невиявленими залишилися 32 відсотки загроз типу ін’єкцій, що стало найгіршим результатом серед усіх інструментів. Nikto може бути ефективним у разі недоліків системи, що пов’язані з заголовками або коли веб-сервери працюють за CGI сценаріями. В такому разі, сканер ухиляючись від системи аналізу вторгнень, може якісно здійснювати огляд захищеності таких систем в автоматичному режимі. Сканер має різноманітні алгоритми налаштування модулів, що може дозволити тестувальнику обрати ефективний підхід, зважаючи на цікавлячу його категорію вразливостей, що є гнучкою й корисною особливістю.

Arachni став кращим інструментом в виявленні вразливостей категорій порушеного контролю доступу, неправильної конфігурації безпеки, помилок в моніторингу та логуванні проблем безпеки. Серед особливостей можна виділити зручне використання гнучкого вбудованого програмного модуля PhantomJS, котрий відіграє роль веб-браузера. В ході прямого порівняння сканерів за виявленнями вразливостей категорії ін’єкцій Arachni показав гарну точність 92 відсотки з відносно невеликою кількістю хибних спрацьовувань. Сканеру невдалося виявити усього лише 3 вразливості, з чого виплаває те, що показник чутливості Arachni склав 88 відсотків. В ході сканування інструмент показав себе не з кращого боку через те, що кожного разу показував відмінну від попередньої спроби сканування кількість виявлених вразливостей. Було виявлено, що така поведінка інструменту напряму залежить від числа надісланих запитів. Задля отримання кращого результату сканування, треба надсилати значну кількість запитів, що провадить вплив на работу сервера. Зважаючи на це, сканер Arachni погано підходить для виконання аналізу клієнт-серверних веб-додатків.

Сканер ZAP є ефективним рішенням для пошуку загроз в клієнт-серверних веб-додатках за умови використання стандартного режиму роботи, а саме сканування за допомогою методу GET. Він гарно проявив себе у знаходженні вразливостей з майже усіх категорій найпоширеніших вразливостей OWASP Top Ten. Точність виявлення ZAP у виявлені вразливостей XSS та SQL дорівнює 100

відсоткам, що стало кращим й нездоланним результатом. Інструменту не вдалося виявити одну вразливість, тому показник чутливості дорівнює 96 відсотків. Сканер зручно використовувати завдяки використанню зручного користувацького інтерфейсу. Маючи нульову кількість хибно-позитивних виявлень, застосування OWASP ZAP в якості інструменту виявлення вразливостей клієнт-серверних веб-додатків є ефективним й бажаним рішенням.

Інструмент виявлення загроз та вразливостей Burp Suite є єдиним платним рішенням з усіх розглянутих сканерів й здатний виконувати динамічне сканування системи. Завдяки застосуванню сканеру було отримано кращі результати виявлень вразливостей у категоріях порушеного контролю доступу, ін’єкцій, вразливих й застарілих компонентій та порушення цілісності даних й ПЗ. В ході однієї зі спроб тестування, Burp просто здійснював спробу реєстрації акаунту, з використанням одних й тих же самих облікових даних, що стало неприємним моментом й тестування прийшлося прервати, але інша спроба все ж таки показала гарний результат виявлення. Сканер показав таку точність виявлень, що є гіршою за сканер ZAP, але все ж таки показник точності дорівнює цілих 96 відсотків, що є гідним результатом. Було знайдено ін’єкцію XML, яка виявилася хибно-позитивною. Burp єдиному вдалося виявити всі вразливості категорій XSS й SQL, тому показник чутливості є 100 відсотків. Хоча Burp і показав помилкове спрацьовування, він все одно може скласти гарну конкуренцію інструменту ZAP, де останній показав кращий результат, принаймні при аналізі виявлений ін'єкцій.

На підставі проведеного аналізу інструментів виявлення вразливостей клієнт-серверного веб-додатку була визначена значна функціональна перевага за кількістю виявлених вразливостей, точністю отриманих результатів й параметром чутливості інструментів Burp Suite й OWASP ZAP. Вони є кращими програмними рішеннями для виявлення загроз клієнт-серверних веб-додатків, як для пересічних користувачів, так і для досвідчених тестувальників.

ВИСНОВКИ

1) Використання методів та практик щодо забезпечення безпеки веб-додатків, так само як і проектування веб-додатків з використанням захищеного дизайну являють собою досить складну задачу. Питання безпеки веб-додатків та їх вразливостей розглядаються багатьма відомими джерелами. У той же час деякими компаніями пропонуються програмні рішення, що дозволяють попередити й уникнути критичних загроз безпеки даних користувачів та застосунку в цілому [30].

2) З ціллю покращення розуміння предметної області методів покращення якості веб-систем та виявлення притаманних їм вразливостей було проведено дослідження галузі створення й тестування веб-додатків, розглянуто базові поняття веб-додатків та їх клієнт-серверної архітектурної моделі, проблем безпеки додатків, розглянуто поняття вразливостей та наведено класифікацію найактуальніших з них та способи боротьби з такими недоліками безпеки веб-систем. Було ознайомлено з відомими методами контролю якості в цілому й з інструментами контролю якості веб-додатків зокрема.

3) При проведенні порівняльного аналізу інструментів та методів контролю якості та пошуку вразливостей у сучасних клієнт-серверних веб-додатках враховувалися такі параметри як: кількість виявлених вразливостей, точніть виявлення вразливостей, хибність виявлення вразливостей, час, сили й навички що необхідні для виявлення проблем безпеки веб-додатків, зручність користування інструментами тощо.

Надано відповідні оцінки ефективості роботи сканерів веб-додатків та розраховано точність отриманих результатів після завершення процесу сканування й обробки знайдених вразливостей й недоліків безпеки аналізованого вразливого клієнт-серверного веб-додатку.

4) Спираючись на проведені дослідження та аналіз було пропрацьовано та сформульовано практичні рекомендації щодо застовування інструментів

забезпечення безпеки й виявлення загроз веб-програм на практиці. Було оцінено доцільність використання сканерів у конкретному випадку в умовах сучасних загроз й вразливостей. Виявлено найкращі з них.

Згідно наданим практичним рекомендаціям щодо застосування інструментів сканування веб-додатків, найбільш доцільними й ефективними з боку отриманих результатів виявилися інструменти Burp Suite та ZAP від OWASP. Ці програмні рішення показали себе найбільш ефективними у виявлені вразливостей. Результати роботи сканерів були найбільш точними попри виявлення великої кількості вразливостей.

5) У роботі отримані значення деяких показників роботи інструментів сканування й отриманих ними результатів, серед яких є: кількість вразливостей, чутливість й точніть отриманих даних. Ці показники можуть відрізнятися від сканування до сканування. Точність отриманих результатів роботи інструментів може бути кращою або гіршою при тестуванні інших клієнт-серверних веб-додатків.

6) Враховуючі можливі похибки у результатах роботи сканерів, рекомендується використовувати комбінації різних програмних рішень, для того щоб мінімізувати хибні результати виявлення вразливостей й отримувати більш цілісні й коректні результати.

7) Враховуючи те, що в ході виконання роботи було проаналізовано, протестовано й порівняно лише безкоштовні інструменти виявлення вразливостей з відкритим кодом або тимчасові програмні рішення, можна припустити що платні інструменти виявлення вразливостей можуть виявитися кращими й більш доцільними рішеннями для контролю якості веб-додатків, адже вони є більш оновлюваними й можуть мати більш широкий функціонал для тестування.

Платні інструменти можуть надавати змогу проводити більш комплексні й обширні тестування веб-додатків, що в свою чергу скажеться на кількості виявлених вразливостей й недоліків систем безпеки програм. Але це не каже про

те, що їх результати будуть більш точними за такі використані нами інструменти контролю якості, як ZAP та BurpSuite, котрі показали гарну точність отриманих результатів.

8) Можливими напрямками майбутніх досліджень є огляд платних програмних рішень контролю якості й пошуку вразливостей веб-додатків. Їх можна використати для виявлення вразливостей у різнотипних веб-додатков з урахування зміни методик й засобів тестування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Яремчук К., Воскобойников Д. Сучасні загрози та методи забезпечення безпеки веб-застосунків. *Комп'ютерні науки та кібербезпека*. 2022. №. 2. С. 26-32. doi: 10.26565/2519-2310-2022-2-03
2. Web application. *Wikipedia The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Web_application (дата звернення: 13.10.2022)
3. Characteristics of Modern Web Applications. *Microsoft*. URL: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/modern-web-applications-characteristics (дата звернення: 13.10.2022)
4. Client Server Architecture – Detailed Explanation. *InterviewBit*. URL: https://www.interviewbit.com/blog/client-server-architecture (дата звернення: 14.10.2022)
5. Client–server model. *Wikipedia The Free Encyclopedia.* URL: https://en.wikipedia.org/wiki/Client–server_model (дата звернення: 16.10.2022)
6. What is the Client-Server Model? *Plain English*. URL: https://javascript.plainenglish.io/what-is-the-client-server-model-6704f4446937 (дата звернення: 16.10.2022)
7. Client-Server Overview. *MDN Web Docs – Mozilla*. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview (дата звернення: 17.10.2022)
8. What is Client Server Architecture? *Intellipaat*. URL: https://intellipaat.com/blog/what-is-client-server-architecture (дата звернення: 17.10.2022)
9. WHAT IS APPLICATION VULNERABILITY? *Contrast Security*. URL: https://www.contrastsecurity.com/glossary/application-vulnerability (дата звернення: 21.10.2022)

10. WhiteHat Security: 50% of Apps Are Vulnerable. *Dark Reading*. URL: https://www.darkreading.com/application-security/whitehat-security-50-of-apps-are-vulnerable (дата звернення: 25.10.2022)
11. OWASP Top Ten. *OWASP Foundation*. URL: https://owasp.org/www-project-top-ten (дата звернення: 25.10.2022)
12. 7 Web Application Security Best Practices You Need to Know. *Cypress Data Defense*. https://www.cypressdatadefense.com/blog/application-security-best-practices (дата звернення: 1.11.2022)
13. 6 BEST PRACTICES FOR QUALITY ASSURANCE TESTING FOR WEB APPLICATIONS. *Unosquare*. URL: https://blog.unosquare.com/6-best-practices-for-quality-assurance-testing-for-web-applications (дата звернення: 3.11.2022)
14. Static application security testing. *Wikipedia The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Static_application_security_testing (дата звернення: 3.11.2022)
15. Dynamic application security testing. *Wikipedia The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Dynamic_application_security_testing (дата звернення: 3.11.2022)
16. Vulnerability Scanners and Scanning Tools: What To Know. *Balbix*. URL: https://www.balbix.com/insights/what-to-know-about-vulnerability-scanning-and-tools (дата звернення: 3.11.2022)
17. Network vulnerability scanning. *TechTarget*. URL: https://www.techtarget.com/searchsecurity/definition/vulnerability-scanning (дата звернення: 6.11.2022)
18. Authenticated vs Unauthenticated Vulnerability Scanning. *Stern Security*. URL: https://www.sternsecurity.com/blog/authenticated-vs-unauthenticated-vulnerability-scanning (дата звернення: 5.11.2022)
19. OWASP Zed Attack Proxy (ZAP). *OWASP Foundation*. URL: https://www.zaproxy.org (дата звернення: 8.11.2022)

20. OWASP ZAP: a powerful tool to discover Websites vulnerabilities. *GURU Advisor*. URL: https://www.guruadvisor.net/en/50-sicurezza/845-owasp-zap-a-powerful-tool-to-discover-websites-vulnerabilities (дата звернення: 8.11.2022)
21. Arachni - Web Application Security Scanner Framework. *GitHub*. URL: https://github.com/Arachni/arachni (дата звернення: 8.11.2022)
22. Arachni Review & Alternatives. *Comparitech*. URL: https://www.comparitech.com/net-admin/arachni-review (дата звернення: 13.11.2022)
23. The web-application vulnerability scanner. *Wapiti*. URL: https://wapiti-scanner.github.io (дата звернення: 14.11.2022)
24. Wapiti – free web-application vulnerability scanner. *Medium*. URL: https://pentestit.medium.com/wapiti-free-web-application-vulnerability-scanner-ce7712adf644 (дата звернення: 15.11.2022)
25. Nikto2. *CIRTt*. URL: https://cirt.net/Nikto2 (дата звернення: 20.11.2022)
26. Nikto Website Scanner. *HackerTarget*. URL: https://hackertarget.com/nikto-website-scanner (дата звернення: 20.11.2022)
27. Burp Suite's web vulnerability scanner. *PortSwigger*. URL: https://portswigger.net/burp/vulnerability-scanner (дата звернення: 20.11.2022)
28. How To Use Burp Suite For Web Application Security Testing. *Software Testing Help*. URL: https://www.softwaretestinghelp.com/how-to-use-burp-suite (дата звернення: 23.11.2022)
29. DAMN VULNERABLE WEB APPLICATION. *GitHub*. URL: https://github.com/digininja/DVWA (дата звернення: 23.11.2022)
30. Inamdar, D. M., & Gupta, S. A Survey on Web Application Security. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2020, 223-228 с.

ДОДАТОК А

СПИСОК ПУБЛІКАЦІЙ МАГІСТРА

ISSN 2519-2310 CS&CS, Issue 2(22) 2022

УДК 681.04

СУЧАСНІ ЗАГРОЗИ ТА СПОСОБИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ

Кирило Яремчук, Денис Воскобойников

Харківський національний університет імені В.Н. Каразіна, Харків, 61022, Україна
kir.yaremchuk@gmail.com, denisvoskobojnykov@gmail.com

Рецензент: Микола Карпінський, д.т.н., проф., університет Бельсько-Біла, Бельсько-Біла, Польща.
mkarpinski@ath.bielsko.pl

Надійшла: Жовтень 2022.

Анотація: *Складність розроблюваних веб-застосунків зростає з кожним роком, що, в свою чергу, робить важкоздійсненим забезпечення їхньої безпеки. Саме тому, доцільно приділяти особливу увагу критичним проблемам захисту програмного забезпечення. Вміння оцінювати ризики та запобігати вразливостям ще на етапі проектування продукту є вкрай важливою задачею, котра знижує потенційні складності при експлуатації застосунку. За останні роки кількість випадків витоку даних у всіх галузях ринку зменшилося, але, їх руйнівність стала значнішою. Серед усіх атак, атаки на веб-застосунки становлять більш ніж 50 відсотків. Згідно зі списком вразливостей OWASP Top Ten, в роботі розглянуто актуальні категорії вразливостей та напрямки атак на існуючі веб-застосунки. Було розглянуто ефективні способи їх запобігання. Наведені рекомендації щодо реалізації та підтримки захищеності додатків, розроблених з використанням бібліотеки ReactJS. Було виділено найпоширеніші загрози безпеки продуктів на базі React на протязі життєвого циклу додатку. Розглянуті основні способи оптимізації ReactJS.*

Ключові слова: *вразливість; веб-застосунки; загрози веб-застосунків; методи безпеки ReactJS*

1. Вступ

Завдяки тривалому часу свого розвитку, веб-застосунки почали являти собою дещо куди значніше, аніж просто сайти з контентом. На просторах мережі Інтернет с кожним днем з'являються все більш складні веб-застосунки за своїми цілями й можливостями, котрі пропонують нові рішення задля задоволення вимог споживачів в усіх галузях ринку. Веб-застосунки являють собою найбільш зручний та ефективний засіб для представлення інформації й надання послуг у мережі. Компанії з різнобічних галузей ринку продовжують створювати веб-застосунки для просування своїх товарів та послуг у Інтернеті, займаючи свої ніші у цифровому світі. Мобільні інструменти та веб-технології захопили вершину списку на довгі роки. Такі цифрові сервіси часто бувають критично значущими й потребують захисту задля забезпечення безпеки різного роду конфіденційної інформації. Галузь веб-технологій розвивається невпинними кроками, однак несе з собою і певні ризики безпеки: - забезпечення захисту значної кількості різнотипної інформації досягти досить складно. Недотримання вимог безпеки може загрожувати втратою ресурсів, а інколи й проблемами з законом [1]. На рис. 1 представлені тенденції злому систем безпеки у 2014-2021 роках [2].

Відповідно звіту *Risk Based Security* про тенденції порушення даних за 2021 рік, у 2020 та 2021 роках було втрачено 27,81 і 18,88 млрд записів, тоді як у 2019 році усього 4,681 млрд. записів [2]. Попри те, що кількість таких випадків зменшилася у 2020 та 2021 роках, втрати даних стали більш масштабними. В цілому, можна спостерігати відсутність кореляції між кількістю випадків втрати даних та кількістю втрачених даних, що говорить про те, що навіть одне малозначне порушення безпеки може спричинити серйозні наслідки. Щоб на практиці знизити можливі ризики безпеки, потрібно ретельно контролювати усі потенційні загрози та уважно дотримуватися існуючих стандартів інформаційної безпеки (ІБ). На рис. 2 представлено огляд основних тенденцій атак (зломів) по галузям економіки у 2021 році [2].

