МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет науки і технологій

Кафедра «Електронні обчислювальні машини»

«ДО ЗАХИСТУ»
Завідувач кафедри
Жуковицький І. В.
(підпис) (ПІБ)
«21» 12 2021 р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань 12 Інформаційні технології
(шифр) (назва)

Спеціальність 123 Комп'ютерна інженерія
(код) (повна назва)

Тема Дослідження систем автоматизованого тестування безпеки веб-додатків

Theme Investigation of systems for automated testing the security of web-documents

Керівник дипломного проекту	д.т.н., професор (посада)	(підпис)	Жуковицький І.В. (ПІБ)
Консультант розділу з БЖД	доцент (посада)	(підпис)	Саблін О. І. (ПІБ)
Нормоконтролер	доцент (посада)	(підпис)	Шановалов В. О. (ПІБ)
Студент групи	КС2021 (група)	(підпис)	Піддубняк П.В. (ПІБ)
Student			Piddubnyak Polina (family name)

Дніпро
2021

Довідка

про відсутність плагіату у випускній кваліфікаційній роботі

Міністерство освіти і науки України

Український державний університет науки і технологій

Кафедра Електронні обчислювальні машини

ДОВІДКА

За результатами перевірки випускної кваліфікаційної роботи здобувача вищої освіти Підгубняк Галіни Володимирівни

(прізвище, ім'я, по батькові)

на тему: Дослідження систем автоматизованого тестування безпеки веб-додатків

в роботі не виявлено порушень академічної доброчесності.

Керівник ВКР Жуковицький І.В.

Український державний університет науки і технологій

Факультет Комп'ютерних технологій і систем кафедра Електронні обчислювальні машини
Спеціальність Комп’ютерна інженерія

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

_____
(підпис)

«___»_____20___р.

ЗАВДАННЯ

до дипломної роботи на здобуття освітнього ступеня «магістр»
(освітнього ступеня)

студента групи КС2021 Піддубняк Поліни Володимирівни
(номер групи) (ПІБ)

1 Тема дипломної роботи Дослідження систем автоматизованого тестування безпеки веб-додатків
затверджена наказом по університету від «31» 08 2021 р. № 508-СТ.

2 Термін подання студентом закінченої роботи_____

3 Вихідні дані до дипломної роботи_____

_____

_____

_____

4 Зміст пояснювальної записки (перелік питань до розробки) _____

_____

_____

_____

_____

5 Перелік креслень (демонстраційного матеріалу)_____

_____

_____

_____

_____

_____

6 Розділи та консультанти

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв

КАЛЕНДАРНИЙ ПЛАН

Назва розділу	Термін виконан-ня	Обсяг розділу, %

Дата видачі завдання: «__»_____20____р.

Керівник дипломної роботи _____ Заєць О.П.

(підпис) (ПІБ)

Завдання прийняв до виконання _____ Піддубняк П.В.

(підпис) (ПІБ)

РЕФЕРАТ

Магістерська робота складається з чотирьох розділів, містить 88 сторінок, 26 рисунків, 5 таблиць, 1 додаток. Перелік використаної літератури містить 21 найменування.

Ключові слова: вразливість, сканування, тестування, веб-додаток, захищеність, атака, ефективність.

Об’єктом дослідження магістерської роботи є процес проектування та дослідження системи автоматизованого тестування веб-додатків.

Мета кваліфікаційної роботи - дослідження та проектування системи автоматизованого тестування безпеки веб-додатків.

У першому розділі роботи досліджено проблеми захисту веб-додатків та визначено актуальність автоматизованого тестування. Другий розділ роботи присвячений огляду засобів захисту веб-додатків. У третьому розділі виконано розробку та дослідження системи автоматизованого тестування безпеки веб-додатків. У четвертому розділі розглянуто питання з охорони праці та безпеки у надзвичайних ситуаціях.

Галузь застосування – розроблену систему автоматизованого тестування безпеки веб-додатків можна застосовувати підприємствами та організаціями, які безпосередньо використовують у своїй діяльності веб-додатки.

Економічна ефективність полягає у зменшенні витрат на купівлю ліцензій автоматизованих систем тестування безпеки веб-додатків.

ABSTRACT

The master's thesis consists of four sections, contains 88 pages, 26 figures, 5 tables, 1 appendix. The list of references literature contains 21 names.

Keywords: vulnerability, scanning, testing, web application, security, attack, effectiveness.

The object of research of the master's thesis is the process of designing and researching a system of automated testing of web applications.

The purpose of the qualification work is to research and design a system of automated security testing of web applications.

The first section examines the problems of web application protection and determines the relevance of automated testing. The second section is devoted to an overview of web application security tools. The third section develops and researches the system of automated security testing of web applications. The fourth section deals with issues of labor protection and safety in emergencies.

Scope - the developed system of automated testing of security of web applications can be applied by the enterprises and the organizations which directly use in the activity web applications.

Cost-effectiveness is to reduce the cost of purchasing licenses for automated web application security testing systems.

ЗМІСТ

ВСТУП

Актуальність теми. В нинішній час веб-додатки набувають все більшої популярності. У зв’язку з цим виникають питання безпеки веб-додатків та дослідження ризиків розкриття конфіденційних даних або важливих бізнес-функцій.

Кібератаки на веб-програми відбуваються щодня, і, на жаль, традиційні брандмауери та інші засоби контролю мережевої безпеки не можуть захистити від багатьох векторів атак, які притаманні веб-додаткам. Тестування безпеки веб-застосунків критично важливо для захисту програм і організації. Веб-програми, стають найбільшим вектором атак для зловмисників, які прагнуть зламати захист. Доступні користувачам цілодобово і без вихідних, веб-додатки - найлегша ціль для хакерів, які прагнуть отримати доступ до конфіденційних внутрішніх даних.

Саме тому актуально застосовувати системи автоматизованого тестування веб-додатків.

Основна мотивація використання систем автоматизованого тестування веб-додатків полягає в тому, що ручна перевірка коду та традиційні плани тестування забирають багато часу, а нові вразливості постійно вводяться або виявляються.

У багатьох областях існують нормативні директиви, які потребують використання інструментів систем автоматизованого тестування безпеки. Більше того – і, можливо, найголовніше – окремі особи та групи, які мають намір зламати системи, теж використовують інструменти, і ті, кому доручено захищати ці системи, повинні «йти в ногу» зі своїми супротивниками.

Використання інструментів систем автоматизованого тестування дає безліч переваг, які збільшують швидкість, ефективність та охоплення для тестування програм.

Тести, які вони проводять, відтворюються і добре масштабуються після того, як тестовий приклад розроблений в інструменті, він може бути виконаний для багатьох рядків коду з невеликими додатковими витратами.

Подібного роду інструменти ефективні при виявленні відомих вразливостей, проблем та слабких місць. Їх також можна використовувати в робочому процесі виправлення, особливо під час перевірки, і їх можна використовувати для кореляції та виявлення тенденцій та закономірностей.

Таким чином, застосування систем автоматизованого тестування безпеки веб-додатків є актуальним задля забезпечення інформаційної безпеки.

На даний час існує досить багато рішень автоматизованого тестування безпеки веб-додатків, але дані рішення є недостатньо ефективними та швидкими.

Окрім цього, в нинішній час широкого застосування набуває використання нейронних мереж з метою прогнозування загроз, але сучасні рішення автоматизованого тестування безпеки веб-додатків не застосовують даний модуль задля підвищення функціональності.

Саме тому в межах роботи буде розглядатися проектування системи, яка перевершить існуючі аналоги у функціональності та можливості прогнозування.

Метою написання магістерської роботи є дослідження та проектування системи автоматизованого тестування безпеки веб-додатків.

При написанні роботи були поставлені наступні задачі:

1. Виконати огляд літератури з теми дослідження. Дослідити проблеми захисту веб-додатків та визначити актуальність автоматизованого тестування;

2. Здійснити огляд засобів захисту веб-додатків та розглянути системи автоматизованого тестування;

3. Виконати розробку та дослідження системи автоматизованого тестування безпеки веб-додатків.

4. Розглянути питання з охорони праці та безпеки у надзвичайних ситуаціях.

Об’єктом дослідження магістерської роботи є процес проектування та дослідження системи автоматизованого тестування веб-додатків.

Предметом дослідження є система автоматизованого тестування безпеки веб-додатків.

При написанні роботи були використані методи аналізу, порівняння, статистики, проектування.

Наукова новизна отриманих результатів полягає у тому, що вдосконалено існуючі системи автоматизованого тестування безпеки веб-додатків з застосуванням модуля прогнозування, яка розроблена на основі використання нейронних мереж.

Практична значимість отриманих результатів полягає в тому, що розроблену систему автоматизованого тестування безпеки веб-додатків можна застосовувати підприємствами та організаціями, які безпосередньо використовують у своїй діяльності веб-додатки.

1 ОГЛЯД ЛІТЕРАТУРИ З ТЕМИ ДОСЛІДЖЕННЯ. ПРОБЛЕМИ ЗАХИСТУ ВЕБ-ДОДАТКІВ ТА АКТУАЛЬНІСТЬ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

1.1 Аналіз статистики атак на веб-додатки. Класифікація атак та вразливостей

В нинішній час велика увага приділяється інформаційній безпеці веб-додатків. Деякі компанії проводять аналіз статистик атак на веб-додатки. В межах даного розділу роботи розглянемо таку статистику, а також наведемо основні типи атак на веб-застосунки.

Як зазначається в [1], основні результати цих досліджень, які характеризують рівень безпеки веб-застосунків показують, що:

- у 77% випадків при роботах із зовнішнього тестування на проникнення було виявлено можливість проведення атак для отримання доступу до внутрішніх корпоративних ресурсів через експлуатацію вразливостей веб-додатків;

- 30% кіберінцидентів пов'язані з атаками на веб-ресурси.

Всі ці дані свідчать про те, що веб-додатки є однією з основних «мішеней» для зловмисників, тому що велика кількість невиправлених уразливостей та простота їх експлуатації допомагають атакуючим успішно досягати своєї мети — від крадіжки інформації до доступу до внутрішніх ресурсів локальної обчислювальної мережі [1].

Важливо розуміти, що більшість уразливостей можна виявити задовго до атаки, а аналіз вихідного коду веб-додатків дозволяє виявити у кілька разів більше критично небезпечних уразливостей, ніж тестування систем без дослідження коду.

Основні результати досліджень, які направлені на аналіз веб-додатків з точки зору інформаційної безпеки, показують, що [1,2]:

1. Усі веб-додатки вразливі За результатами автоматизованого аналізу вихідного коду було встановлено, що всі веб-додатки мають уразливості, причому лише у 6% досліджених систем відсутні вразливості високого ступеня ризику.

2. Користувачі веб-додатків - основна мета У 85% протестованих веб-застосунків присутні вразливості, які дозволяють проводити атаки на користувачів. Використовуючи дані вразливості, зловмисник може викрадати cookie користувачів, проводити атаки фішинга або заражати їх робочі станції шкідливим програмним забезпеченням.

3. Веб-додатки фінансових організацій найбільш уразливі У всіх протестованих додатках банків та інших фінансових організацій знайдено вразливість високого ступеня ризику. Подібні результати пов'язані з тим, що програми банків мають більш складну логіку роботи в порівнянні з інформаційними веб-ресурсами інших організацій, і цей фактор створює передумови для появи більшої кількості вразливостей високого ступеня ризику. Внаслідок експлуатації даних уразливостей зловмисник може спробувати порушити працездатність програми або виконати довільний код на цільовій системі, що може призвести до повного контролю над сервером із веб-додатком.

4. Усі протестовані веб-додатки державних установ можуть використовуватись для атак на користувачів. У всіх досліджених веб-додатках державних установ є вразливості, що дозволяють проводити атаки на користувачів. При цьому важливо відзначити, що переважна більшість користувачів таких веб-ресурсів дуже погано обізнані з інформаційною безпекою і легко можуть стати жертвами зловмисників.

5. Відмова в обслуговуванні - найпоширеніша загроза для інтернет-магазинів. У 75% досліджених веб-додатків, пов'язаних з електронною торгівлею, виявлено вразливості, які потенційно призводять до відмови в обслуговуванні. Реалізація цієї загрози може призвести до значних фінансових втрат для власників [1,2].

Хакерська атака – це комплекс дій, спрямованих на пошук уразливостей у цифрових системах. При цьому слід зазначити, що хакери не завжди займаються шкідливою діяльністю, проте сьогодні термін «хакерство» зазвичай вживається у контексті протиправних дій, а хакерами називають кіберзлочинців, які прагнуть отримати фінансову вигоду, висловити протест, зібрати певну інформацію [3].

Автоматизований аналіз захищеності, наведений в [1] показав, що всі протестовані веб-додатки містять вразливість різного ступеня ризику. При класифікації вразливостей за рівнем ризику було встановлено, що більшість їх (65%) належить до середнього рівня небезпеки.

Розподіл вразливостей за рівнем ризику наведений на рисунку 1.1.

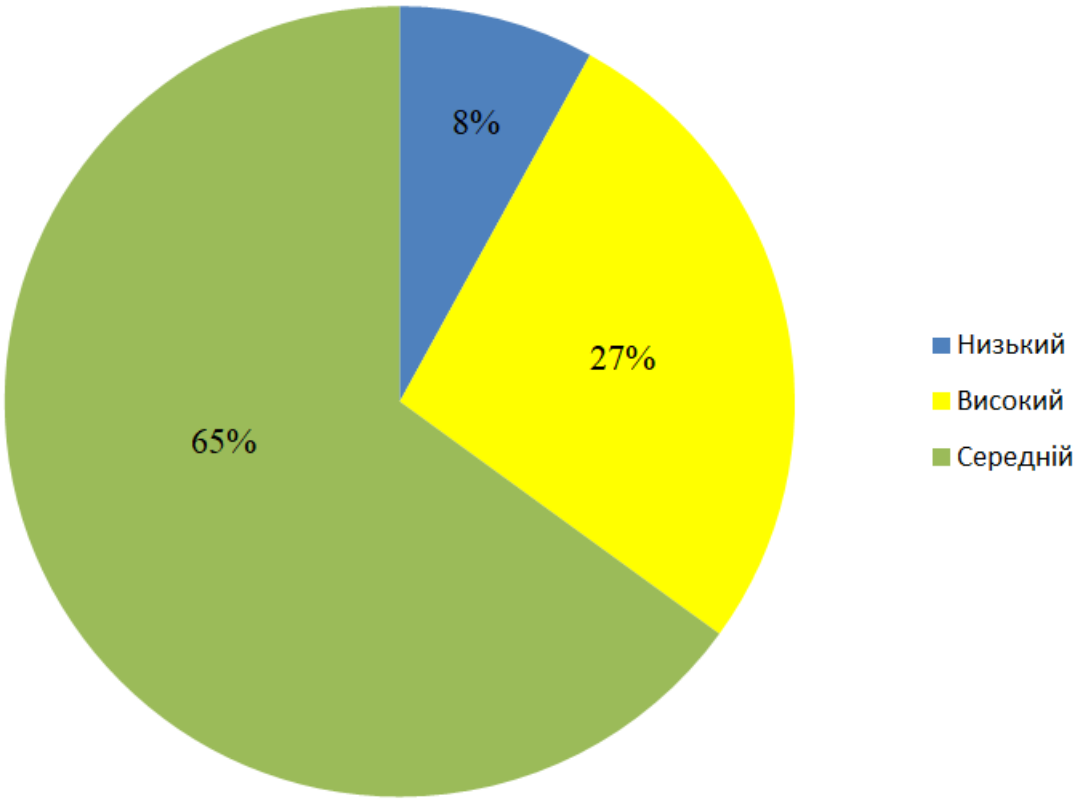

Рисунок 1.1 - Розподіл вразливостей за рівнем ризику

При оцінці систем за максимальним рівнем ризику виявлених уразливостей було встановлено, що лише у 6% досліджених веб-додатків відсутні вразливості високого рівня ризику. Слід враховувати, що досліджувані додатки є типовими CMS-платформами і містять велику

кількість коду, написаного їх власниками. Але при цьому не слід забувати, що успішна експлуатація навіть однієї критично небезпечної вразливості на практиці може призвести до повної компрометації програми або навіть сервера (рисунок 1.2).

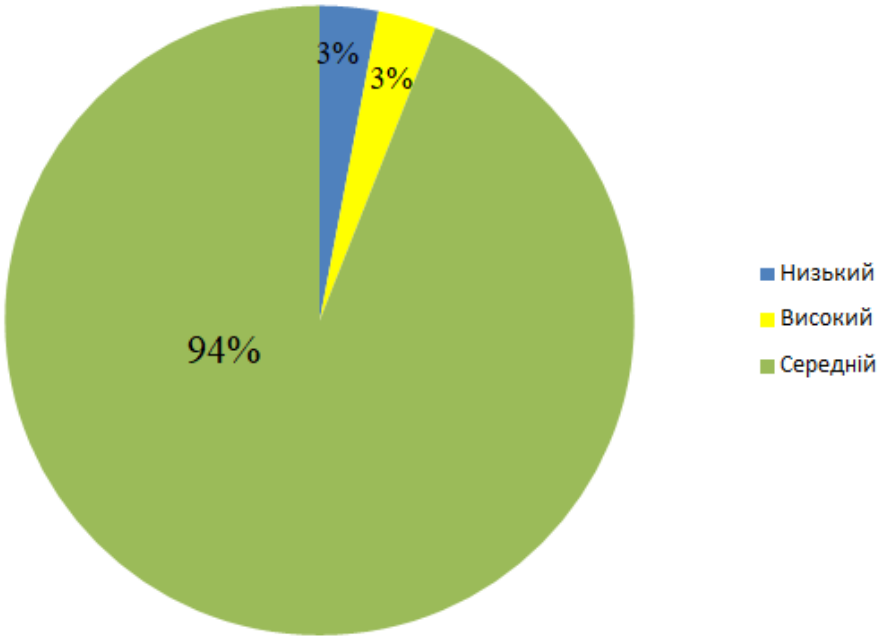

Рисунок 1.2 - Розподіл систем за максимальним рівнем ризику виявлених уразливостей

Найпоширеніша вразливість, що виявилася при автоматизованому аналізі вихідного коду додатків та наведена в аналізі [1] — «Міжсайтове виконання сценаріїв», за допомогою якої зловмисник може проводити фішингові атаки на клієнтів веб-додатків або заражати їхні робочі станції шкідливим програмним забезпеченням. Ця вразливість також лідирує і в аналогічному рейтингу, що базується на результатах ручного тестування веб-додатків.

Друга за поширеністю вразливість — «Розщеплення HTTP відповіді», при успішній експлуатації якої зловмисник може проводити атаки на клієнтів веб-програми, засновані на відправці браузеру подвійної HTTP відповіді, що атакується, вміст заголовків і полів якого частково контролюється порушником.

На третьому місці - вразливість високого рівня ризику "Читання довільних файлів", за допомогою якої зловмисник може отримати несанкціонований доступ до вмісту довільного файлу на сервері. Таким чином, порушник може отримати вихідний код веб-додатка, облікові дані та іншу інформацію, що обробляється в системі.

З десяти найпоширеніших уразливостей п'ять мають високий рівень ризику, і їх експлуатація може призвести до серйозних наслідків. Наприклад, у деяких випадках внаслідок експлуатації вразливості «Створення довільного файлу» зловмисник може виконати довільний код на цільовій системі та надалі повністю скомпрометувати сервер, що атакується [1].

Класифікацією вразливостей веб-додатків також займається Консорціум з безпеки веб-додатків WASC (Web Application Security Consortium).

Консорціумом WASC розроблено спеціальний документ, в якому описано класифікацію вразливостей веб-додатків – WASC Threat Classification [4].

Згідно з документом WASC Threat Classification, уразливості веб-додатків діляться за наступними етапами життєвого циклу програмного забезпечення:

- етап проектування - охоплює вразливості, які можуть виникнути внаслідок помилок у проектуванні веб-додатку;

- етап реалізації - охоплює вразливості, які можуть виникнути через помилки при реалізації компонентів веб-додатку, наприклад – написання програмного коду

- етап розгортання - охоплює вразливості, які можуть виникнути під час налаштування веб-додатку до роботи, наприклад, - неправильна конфігурація веб-сервера;

Помилки, які можуть бути допущені на різних етапах життєвого циклу програмного забезпечення, описані у списку CWE (Common weakness enumeration), сформованого корпорацією MITRE. Згідно з даними веб-сайту

корпорації MITRE, CWE є базовим стандартом при ідентифікації слабких місць у програмному забезпеченні та запобігання їх появі [5].

Кожному недоліку до списку надано власний ідентифікатор (ID), список містить близько тисячі CWE ID. При описі вразливості бази даних уразливостей може бути вказано CWE ID для додаткової інформації.

Остання редакція документа розглядає такі типи атак на веб-додатки [6]:

- вставка інструкцій (Injection) - відбувається вставка SQL - інструкцій, що передаються на обробку веб-додатку, яка після їх отримання може почати виконувати довільні команди;

- некоректна аутентифікація (Broken Authentication) - функції аутентифікації можуть бути реалізовані таким чином, що дозволяють оминати паролі або отримувати ідентифікатори користувачів;

- витік критичних даних (Sensitive Data Exposure) - недостатня захищеність персональних даних;

- атаки на засоби аналізу XML-введення (XML Eternal Entities) – зловмисники можуть завантажувати XML-файли на сервер або включати шкідливий код у документ XML;

- неправильний контроль доступу (Broken Access Control) - контроль доступу реалізовано таким чином, що авторизовані користувачі можуть мати у системі такі повноваження, які повинні мати;

- небезпечна конфігурація оточення (Security Misconfiguration) – відсутність оновлень та неправильна конфігурація окремих компонентів веб – додатків може нести додаткову загрозу безпеці;

- міжсайтове виконання сценаріїв (XSS) - зловмисник отримує можливість виконувати сценарії у браузері жертви, перехоплювати сценарії користувача, перенаправляти користувачів на інші веб-сайти;

- незахищений процес десеріалізації (Insecure Deserialization) – зловмисник може порушувати логіку роботи веб-програми, підробляючи

об'єкти програми, що призводить до віддаленого виконання коду зловмисника;

- використання компонентів з відомими вразливостями (Using Components with Known Vulnerabilities) - програмне забезпечення, у якого термін підтримки вже вичерпано, або яке є не оновленим, що може залучити більше зловмисників до його злому;

- недостатній моніторинг та ведення журналів подій (Insufficient Logging and Monitoring) - погано організований механізм ведення журналів подій та моніторингу може призвести до того, що зловмисники можуть неодноразово робити атаки на веб-додатки, залишаючись непоміченими.

Згідно з дослідженнями компанії Vercode, яка займається тестуванням захищеності програмного забезпечення, при проведенні тестування на проникнення в 74% програм знаходиться як мінімум одна вразливість зі списку OWASP Top – 10 [7].

Подаємо нижче опис кожної з вразливостей.

1) Вставлення інструкцій. Прикладом вставлення інструкцій може бути вставка SQL - інструкцій.

Атака типу "вставка SQL - інструкцій" може бути реалізована через некоректну обробку вхідних даних, що використовуються в SQL - запитах.

Далі виконано аналіз прикладу здійснення атаки типу вставки SQL - інструкцій.

Якщо існує серверне програмне забезпечення, яке отримує вхідний параметр id, використовує його для створення SQL-запиту, то PHP - скрипт обробки запиту може мати такий вигляд:

$Id = $_REQUEST['id'];

$ Res = mysqli_query («SELECT * FROM news WHERE id_news =«. $ Id);

Якщо на сервер буде передано параметр id, що дорівнює 5 (http://example.org/script.php?id=5), то виконається наступний SQL-запит:

SELECT * FROM news WHERE id_news = 5

Але якщо зловмисник передасть id рядок -1 OR 1 = 1 (http://example.org/script.php?id=-1+OR+1=1), виконається запит:

SELECT * FROM news WHERE id_news = -1 OR 1 = 1

Отже, зміна вхідних параметрів шляхом додавання до них конструкцій мови SQL викликає зміну в логіці виконання SQL-запиту (у прикладі вище замість запису із заданим ідентифікатором будуть обрані всі наявні в базі запису).

На рисунку 1.3 зображено інтерфейс тестового веб-додатку, що дозволяє користувачам виводити списки транзакцій, здійснених за певний період [7].

Введення наступного значення дозволяє виконати команду, яка повертає вміст бази даних, включаючи імена та паролі користувачів: 1/1/2010 union select userid, null, username + " + password, null from users--.

Рисунок 1.3 – Інтерфейс тестової веб-програми

Згідно з рисунком 1.3, перша частина значення, введеного в поле дати, є датою - 1/1/2010. Ці дані обробляються веб-додатком. За датою слідує оператор union, що означає початок SQL - команди. Методом спроб та помилок зловмисник може обчислити назву таблиці, що містить дані про облікові записи користувачів (вона називається users), та кількість полів у ній.

Після цього зловмисник використовує цю інформацію для складання SQL - команди select, що повертає дані з таблиці користувачів. Отримана інформація, що містить імена та паролі облікових записів користувачів, відображається на веб-сторінці у формі для виведення списку транзакцій [7].

2) Некоректна автентифікація. Згідно з дослідженнями, зловмисники можуть мати доступ до сотень мільйонів допустимих комбінацій імені користувача та паролю для заповнення форм у веб-додатках, адміністративних списках облікових записів, інструментах перебору значень за словником [7].

Зловмисникам достатньо отримати доступ лише до кількох облікових записів або лише одного облікового запису адміністратора, щоб зламати систему, вкрасти гроші, використовувати методи шахрайства, викрасти конфіденційну інформацію та ін.

Атаки на засоби аутентифікації можливі, якщо веб-додаток має такі слабкі місця [7]:

- місця для заповнення облікових даних, коли зловмисник має список допустимих імен користувачів та паролів;

- відсутній захист від реалізації атак методом перебору;

- дозволено прості або відомі паролі, такі як «password» або «admin/admin»;

- використовуються неефективні процеси відновлення облікових даних та забутих паролів, які не можуть бути безпечними;

- відсутня багатофакторна автентифікація;

- надання ідентифікаторів сеансу в URL-адресі.

Прикладом атаки на засоби автентифікації може бути випадок, коли зловмисник знає секретний ідентифікатор сеансу і може представитися веб-серверу автентифікованим користувачем та скомпрометувати обліковий запис.

На рисунку 1.4 зображено приклад тестового веб-додатку, де показаний результат виконання скрипту для отримання ідентифікатора сеансу [7].

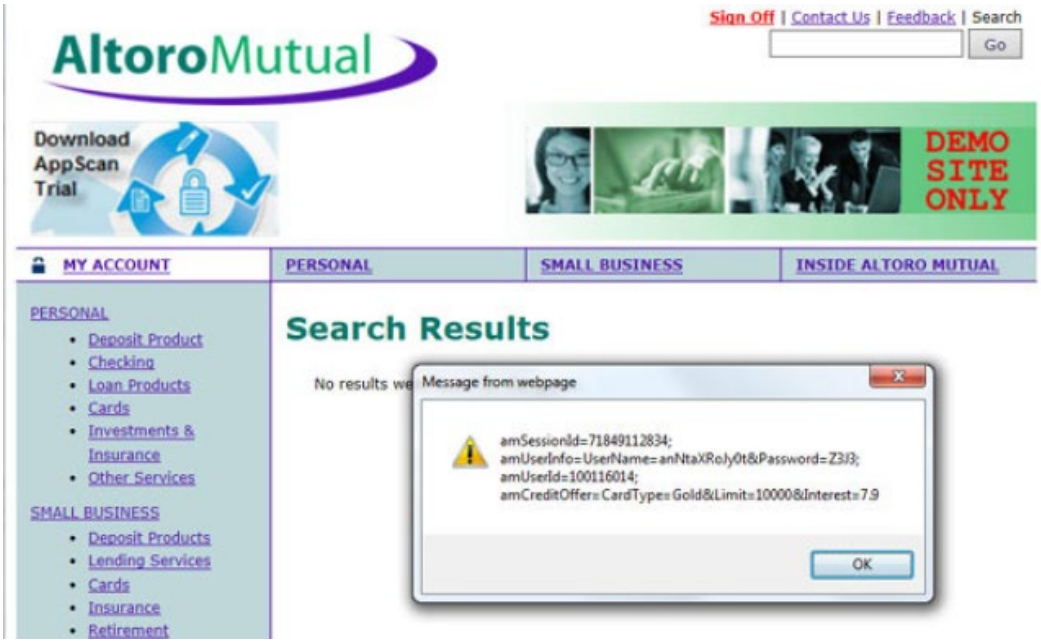

Рисунок 1.4 – Приклад роботи тестового скрипту

3) Витік критичних даних. Згідно з дослідженнями, протягом останніх кількох років викрадення критичних даних було поширеною атакою. Головним недоліком при цьому залишається відсутність шифрування конфіденційних даних або коли використовуються нестійкі до злому криптографічні алгоритми, засоби генерації ключів та управління [7].

Для визначення можливості витоку критичних даних треба визначити, які дані потребують захисту в кожному конкретному випадку при їх передачі та зберіганні. Такими даними можуть бути паролі, номери кредитних карток, медичні записи, особиста інформація та ділова документація або конфіденційна інформація. Для всіх цих типів даних необхідно перевірити [7]:

- чи є дані, що передаються у вигляді відкритого тексту (це стосується таких протоколів, як HTTP, SMTP та FTP);

- чи використовуються старі чи нестійкі до злому криптографічні алгоритми у системі чи коді;

- як реалізований механізм розподілу та управління криптографічними ключами;

- чи використовується поштовим клієнтом недійсний сертифікат, отриманий від сервера.

Одним із сценаріїв витоків важливих даних може бути процес шифрування програмою номерів кредитних карток у базі даних за допомогою автоматичного шифрування бази даних. Однак ці дані автоматично розшифровуються під час завантаження, дозволяючи застосовувати вставки SQL - інструкцій для отримання номерів кредитних карток у вигляді відкритого тексту.

Іншим сценарієм витоку критичних даних може бути веб-додаток, де не використовується протокол захисту транспортного рівня TLS (Transport Layer Security) для всіх сторінок або використовуються нестійкі до злому криптографічні алгоритми шифрування [7].

Зловмисник може здійснити моніторинг мережного трафіку (наприклад, у незахищеній бездротовій мережі), знизити рівень зв'язків з HTTPS до рівня HTTP, перехопити запити та викрасти дані з cookie користувача. Нападник відтворює цей файл cookie і перехоплює сеанс користувача, отримуючи доступ або змінюючи особисті дані.

Крім того, таким чином може змінюватись одержувач грошового переказу. Також до витоку критичних даних може спричиняти використання простих алгоритмів обчислення хеша пароля.

У такому разі всі хеші можуть бути обчислені за допомогою райдужних таблиць попередніх хешів за допомогою графічних процесорів [7].

4) Атаки на засоби аналізу XML – введення XML (eXtensible Markup Language) – це стандарт для обміну структурованими даними у текстовому форматі.

XML-файлу може мати такий вигляд [7]:

version = "1.0" encoding = "UTF-8"?>

<Order>

<Product>1234</product>

<Count>1</ count>

<Orderer>

<Contact> Jan P. Monsch</contact>

<Account>789</account>

</Orderer>

</Order>

Можливі цілі при здійсненні атаки:

- Мережевий сервіс;

- генератор XML;

- XML-аналізатор;

- Код програми.

Нижче наведені можливі приклади організації атак засобами аналізу XML - введення.

На рисунку 1.5 наведено приклад визначення типу даних, який називається foo, з елементом bar, з яким пов'язане слово «World» [7].

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar "World">
]>
<foo>
  Hello &bar;
</foo>
```

Response

```
HTTP/1.0 200 OK

Hello World
```

Рисунок 1.5 – Приклад передачі даних XML-елементів до аналізатора

Згідно з дослідженнями, XML - об'єкти можуть використовуватися зловмисниками для атак "Відмова в обслуговуванні" шляхом вставки об'єктів з елементами, всередині яких знаходяться інші елементи тощо. На рисунку 1.6 представлений приклад запиту до XML – аналізатора та результат обробки запиту.

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar "World ">
  <!ENTITY t1 "&bar;&bar;">
  <!ENTITY t2 "&t1;&t1;&t1;&t1;">
  <!ENTITY t3 "&t2;&t2;&t2;&t2;&t2;">
]>
<foo>
  Hello &t3;
</foo>
```

Response

```
HTTP/1.0 200 OK

Hello World World World World World World Wor
ld World World World World World World World
World World World World World World World Wor
ld World World World World World World World
World World World World World World World Wor
ld World World World
```

Рисунок 1.6 - Приклад передачі до XML - аналізатору об'єкта з вкладеними елементами та результат обробки запиту

Надсилаючи необхідні запити до XML-аналізатора, зловмисник, який володіє інформацією про структуру веб-додатка, може дізнатися інформацію про програмне забезпечення, яке використовується [7].

Приклад можливих інструкцій у XML – файлі та результат роботи XML – аналізатора наведено на рисунку 1.7.

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar SYSTEM
  "file:///etc/lsb-release">
]>
<foo>
  &bar;
</foo>
```

Response

```
HTTP/1.0 200 OK

DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04 LTS"
```

Рисунок 1.7 - Приклад передачі XML - аналізатору об'єкта з інструкціями визначення операційної системи

5) Неправильний контроль доступу. Призначення контролю доступу – забезпечити виконання вимог політики безпеки таким чином, щоб користувачі системи не змогли виконувати у системі дії, що виходять за межі повноважень користувачів. Прикладом контролю доступу може бути обмеження доступу до ресурсів неавторизованих користувачів [7].

Однак, як відомо, обмеження доступу до ресурсів може стосуватися не тільки файлів та баз даних, а також:

- програм;
- спеціальних функцій додатків;
- спеціальних полів даних;
- пам'яті;
- приватних чи захищених змінних
- носіїв інформації;
- засобів передачі інформації.

Недоліки в засобах контролю доступу зазвичай призводять до несанкціонованого розкриття інформації, модифікації або знищення всіх даних або виконання бізнес-функцій за межами користувача. Загальні уразливості засобів контролю доступу включають:

- обхід перевірок контролю доступу шляхом зміни URL-адреси, внутрішнього стану програми, HTML-сторінки або просто використання спеціального інструменту атаки на програмний інтерфейс;

- можливість за допомогою первинного ключа змінювати всі записи інших користувачів, що дозволяє переглядати та редагувати чужі облікові записи;

- підвищення привілеїв (повноваження користувача у системі без проходження етапів авторизації, або повноваження адміністратора після входу до системи як користувача;

- маніпулювання метаданими, такими як повторне використання або заміна токенів керування доступом або файлів cookie, за допомогою яких можна підвищувати привілеї в системі.

6) Небезпечна зміна оточення. Згідно з даними досліджень організації OWASP, можливість неправильної конфігурації оточення, тобто середовища функціонування веб-додатку, може статися на будь-якому рівні компонентів програми, включаючи мережеві сервіси, платформу, веб-сервер, сервер додатків, базу даних та ін.

Автоматизовані сканери вразливості можуть виявляти неправильні конфігурації, використовувати облікові записи або стандартні конфігурації.

Доступність автоматизованих засобів пошуку вразливостей допомагає зловмисникам здійснити несанкціонований доступ до деяких системних даних або функціональних можливостей. Іноді такі недоліки призводять до повного розкриття структури системи та викрадення конфіденційних даних.

Додаткову загрозу несе наявність програмного забезпечення, яке не оновлюється [7].

7) Міжсайтове виконання сценаріїв. Атаки на веб-системи, в яких уразливість полягає в тому, що виконується вставка коду зловмисника в сторінку веб-програми, яка буде виконана на комп'ютері користувача при відкритті ним цієї сторінки і відбувається взаємодія цього коду з веб-сервером зловмисника.

Даний тип атаки відомий як XSS – атаки.

8) Незахищена десеріалізація. Процес серіалізації використовується для передачі об'єктів по мережі та збереження їх у файлах. Для цього об'єкт заповнюється потрібними даними, потім викликається код серилізації, і результатом є XML - документ або документ іншого формату.

Результат серіалізації передається стороною, що приймає, по електронній пошті або HTTP. Одержувач створює об'єкт того самого типу, який був до серіалізації, тим самим проводячи процес десеріалізації.

9) Використання компонентів із відомою вразливістю. У структурі веб-програми використовується багато компонентів, виробники яких можуть відрізнятися. По-перше, при використанні різних компонентів при побудові веб-додатки слід враховувати, що при створенні компонентів на кожному з

можливих етапів виробником можуть бути допущені помилки, які можуть призводити до виникнення вразливостей цих компонентів. Компонентами веб-додатків можуть бути реалізації різних сервісів, сторонні бібліотеки, системи керування контентом, реалізації веб-серверів, операційні системи тощо [7].

По-друге, якщо вразливість компонента існує, її можна знайти виробником компонента, користувачем чи зловмисником. Завдання виробника полягає у виправленні вразливості. Але виправлення вразливості може зайняти деякий час, який можуть використовувати зловмисники для атак на користувачів програм, які містять компоненти з уразливістю.

10) Недостатній моніторинг та ведення журналів подій. Як відомо, використання зловмисниками можливої недостатньої реалізації процесу моніторингу та ведення журналів подій є основою майже всіх основних інцидентів. Однією зі стратегій визначення того, чи достатній рівень моніторингу реалізований - це перевіряти журнали реєстрації подій після проведення тестування на проникнення.

До недоліків безпеки програми належать невстановлені або некоректно налаштовані значення різних властивостей та директив. Наприклад, у деяких додатках не було встановлено властивості requireSSL, що відповідає за встановлення прапора secure для HTTP заголовка Set-Cookie, який визначає необхідність передачі cookie тільки за захищеним з'єднанням (HTTPS). Недоліки такого роду відповідно до класифікації уразливостей OWASP6 відносяться до категорії A5 - Security Misconfiguration.

Розподіл виявлених уразливостей відповідно до класифікації OWASP Top 10 наведено на рисунку 1.8. Усі вразливості, які не входять до списку 10 найнебезпечніших, були поміщені до категорії «Інші» [1].

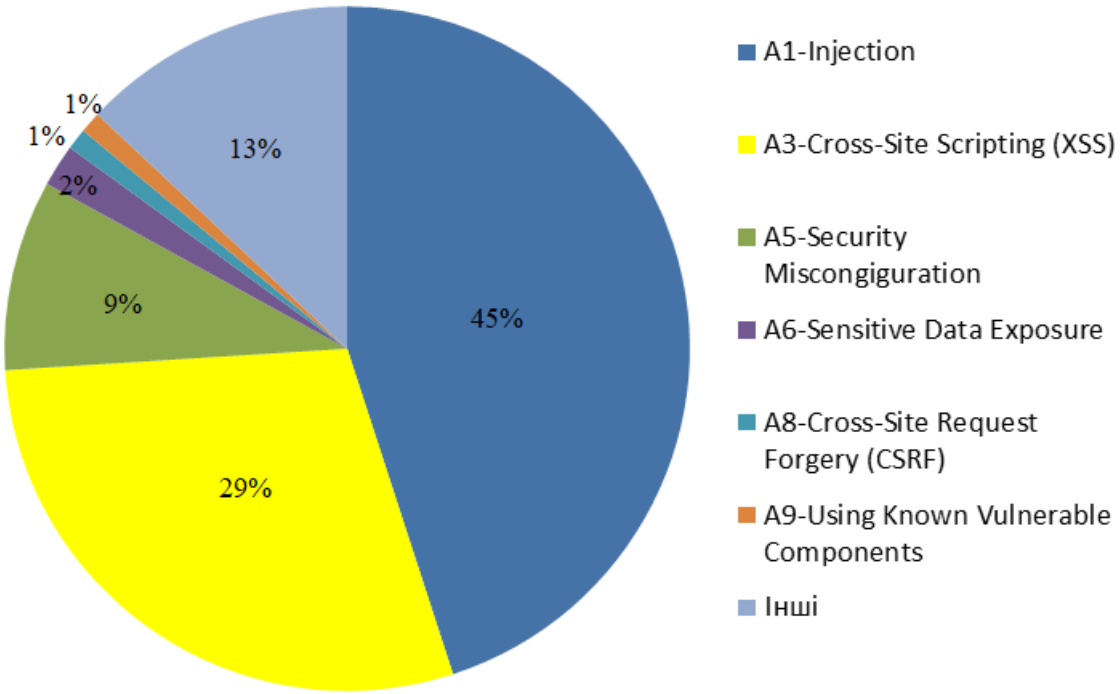

Рисунок 1.8 - Розподіл виявлених уразливостей відповідно до класифікації OWASP Top 10 [1]

Проте в 2021 році, зважаючи на світову ситуацію, в огляді OWASP Top 10 відбулися певні зміни. Дані зміни наведені на рисунку 1.9.

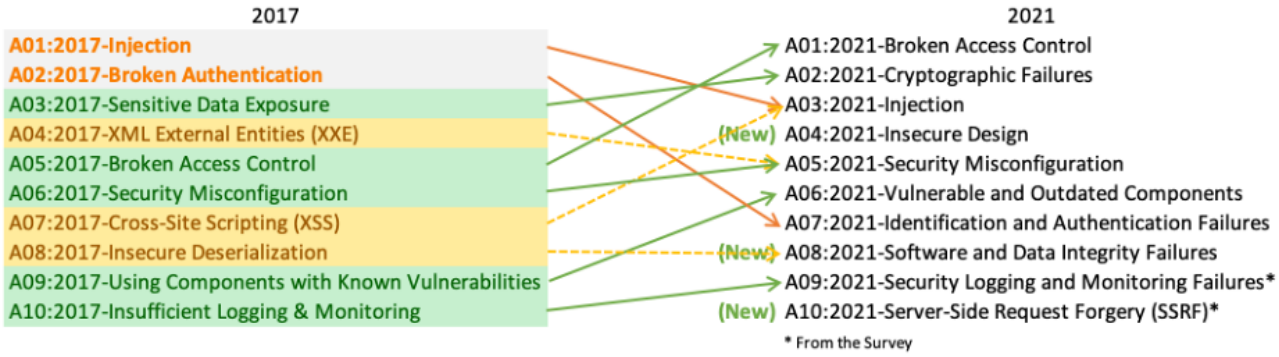

Рисунок 1.9 – Зміни в огляді вразливостей веб-додатків OWASP Top 10

A01: 2021-Broken Access Control переміщується з п'ятої позиції до категорії з найбільш серйозною загрозою безпеці веб-додатків; дані показують, що в середньому 3,81% протестованих додатків мали один або кілька загальних переліків слабких місць (CWE) з більш ніж 318 000 випадків CWE у цій категорії ризику. 34 CWE, зіставлені з порушеним контролем доступу, частіше зустрічалися у додатках, ніж будь-яка інша категорія.

A02: 2021-Cryptographic Failures переміщається на одну позицію до позиції 2, раніше відомої як A3: 2017-Sensitive Data Exposure. В оновленій назві основна увага приділяється збоям, пов'язаним із криптографією, як це було неявно раніше. Ця категорія часто призводить до розкриття конфіденційних даних або компрометації системи.

A03: 2021-ін'єкція опускається на третю позицію. 94% додатків були протестовані на ті чи інші форми ін'єкцій з максимальною частотою 19%, середньою частотою 3,37%, а 33 CWE, віднесені до цієї категорії, посідають друге місце за частотою народження додатків з 274 тис. випадків. Міжсайтові сценарії тепер є частиною цієї категорії у новій редакції від 2021 року.

A04: 2021-Небезпечний дизайн - це нова категорія на 2021 рік, в якій основна увага приділяється ризикам, пов'язаним із недоліками дизайну. Небезпечний дизайн не може бути виправлений за допомогою ідеальної реалізації, оскільки за визначенням необхідних заходів безпеки ніколи не створювалися для захисту від конкретних атак.

A05: 2021-Неправильна конфігурація безпеки переміщається з № 6 у попередній редакції; 90% додатків були протестовані на наявність неправильної конфігурації в тій чи іншій формі, із середньою частотою 4,5%, і більше 208 тисяч випадків CWE були зіставлені з цією категорією ризику. Не дивно, що завдяки більшій кількості переходів у програмне забезпечення, що легко налаштовується, ця категорія просувається вгору. Колишня категорія для зовнішніх об'єктів A4: 2017-XML (XXE) є частиною цієї категорії ризику.

A06: 2021-Вразливі та застарілі компоненти. Ця категорія піднялася з 9-го місця у 2017 році та є відомою проблемою. Це єдина категорія, в якій загальні вразливості та впливу (CVE) не зіставлені з включеними CWE, тому їх бали враховуються експлойт за умовчанням і вага впливу 5,0.

A07: 2021-Збої ідентифікації та аутентифікації раніше називалися порушеною аутентифікацією та поступово знизилися з другої позиції і тепер включають CWE, які більше пов'язані з помилками ідентифікації. Ця

категорія, як і раніше, є невід'ємною частиною Топ-10, але, схоже, допомагає збільшена доступність стандартизованих фреймворків [8].

A08: 2021-Збої цілісності програмного забезпечення та даних - це нова категорія на 2021 рік, в якій основна увага приділяється припущенням, пов'язаним з оновленнями програмного забезпечення, критично важливими даними та конвеєрами CI/CD без перевірки цілісності. Один з найбільш виважених впливів від даних Common Vulnerability and Exposures / Common Vulnerability Scoring System (CVE/CVSS), зіставлених з 10 CWE у цій категорії. A8: 2017-Небезпечна десеріалізація тепер є частиною цієї більш широкої категорії.

A09: 2021-Security Logging and Monitoring Failures раніше був A10: 2017-Недостатнє ведення журналу та моніторинг, що додане з опитування спільноти Top 10 (# 3), піднявшись з # 10 раніше. Ця категорія розширена за рахунок включення більшої кількості типів збоїв, її складно перевірити, і вона погано представлена в даних CVE/CVSS. Однак збої в цій категорії можуть безпосередньо вплинути на видимість, оповіщення про інциденти та криміналістичну експертизу.

A10: Підробка запитів на стороні сервера 2021 додана з опитування спільноти Top 10 (# 1) [8].

У 80% досліджених програм фінансових організацій виявлено вразливість «Міжсайтове виконання сценаріїв», і майже в половині — вразливість «Розщеплення HTTP-відповіді». Саме через ці вразливості в 87% програм можливе проведення атак на користувачів веб-додатків. Загроза відмови в обслуговуванні пов'язана з можливістю успішної експлуатації вразливостей високого ступеня ризику "Впровадження зовнішніх сутностей XML" та "Зміна довільних файлів".

Крім того, за допомогою вразливості "Зміна довільних файлів" зловмисник може спробувати виконати довільний код на цільовій системі, що дасть йому повний контроль над сервером із веб-додатком.

1.2 Проблеми захисту веб-додатків

У міру розвитку технології веб-додатків слід дотримуватися і надійних заходів безпеки. Загрози безпеки веб-застосунків реальні і відбуваються по всьому світу. Стандартних заходів вже недостатньо для захисту від загроз, що розвиваються.

З розвитком технологій постійно з'являються нові вектори атак. Зараз як ніколи потрібні комплексні інструменти безпеки. Раніше захист кінцевих точок пристроїв та мережна безпека були остаточними заходами захисту. Потім були мобільні та хмарні технології, що значно знизили ефективність мережної безпеки і суттєво зруйнували підхід захисту, орієнтований на горизонт, на який так часто покладалися.

Нині еволюціонує зовсім новий підхід до доступу до систем бек-офісу, відкриваючи нові можливості для бізнесу - як законні, так і незаконні. В даний час організації все більше покладаються на інтерфейси прикладного програмування (API) для стимулювання інновацій, швидкість розробки та надання нових можливостей монетизації.

Але, згідно з OWASP Топ-10: «За своєю природою API-інтерфейси розкривають логіку програми та конфіденційні дані, такі як особиста інформація (PII), і через це API-інтерфейси все частіше стають мішенню для зловмисників». Перенесення важливих компонентів на клієнтську сторону додатків (тобто за межами захисту традиційної мережевої безпеки) створює нову масивну поверхню атаки, збільшуючи ризик атак, таких як зламування форми, підробка об'єктної моделі документа (DOM), зловживання сеансом, накладання атаки та зловживання API [9].

На жаль, багато організацій, як і раніше, вважають, що брандмауер веб-застосунків (WAF) - це все, що потрібно для захисту веб-додатків від загроз безпеки. Насправді,WAF - це лише частина рішення. Традиційні методи безпеки, такі як WAF, не можуть зупинити сьогоднішні атаки на веб-застосунки, які в багатьох випадках виходять з браузера поза сферою дії

мережної безпеки. На той час, коли вони спрацьовують – якщо взагалі будь-коли – у зловмисника вже немає конфіденційної інформації, і все, що можна зробити, – це зосередити зусилля на боротьбі зі шкодою.

Оскільки зловмисники продовжують виявляти нові вектори атак, організації повинні відповідати тим самим, щоб забезпечити належний та повний захист критично важливих активів та даних.

Приблизно 95% веб-сайтів працюють на JavaScript та HTML5 - мовах, які можна легко перехопити, переглянути та зламати. Це залишає веб-застосунки та API-інтерфейси вразливими для атак на стороні клієнта, особливо якщо вони покладаються тільки на традиційні інструменти захисту периметра, такі як WAF. За даними Symantec, понад 4800 веб-сайтів щомісяця зазнають злому. Атаки з метою крадіжки даних на стороні клієнта - це лише один із векторів загроз, з якими стикаються веб-додатки, тому важливим є багаторівневий підхід до безпеки [10].

Як видно з недавніх зламів веб-додатків на British Airways та Ticketmaster, навіть якби WAF був на місці та правильно налаштований, він не зміг би запобігти цим порушенням на стороні браузера / клієнта. Як тільки експлойти були виявлені, сотні тисяч клієнтських записів вже були витягнуті.

Таким чином, безпека – ключова особливість розробки сучасних веб-додатків. Щоб залишатися конкурентоспроможними на ринку, компанії повинні пропонувати нові рішення безпеки, щоб протистояти хакерам та надавати своїм клієнтам надійні та безпечні програми.

Проте більшість безпеки веб-додатків залежить від обізнаності розробників про кіберзагрози та запланованого моніторингу дій програми.

1.3 Ієрархія захисту веб-додатків

Для кожного виду інформації необхідний рівень захисту. Сьогодні ієрархію захисту веб-додатків, згідно [10], ділять на шість рівнів.

Перший - найпростіший і обов'язковий. Тут головний інструмент захисту – firewall. Firewall має лімітувати використання сервісів, що надаються користувачам. Також firewall повинен стежити за всіма з'єднаннями як з одного, так і з іншого боку. Не варто застосовувати програми незрозумілого походження, треба віддавати перевагу ліцензійному ПЗ.

Другий рівень безпеки передбачає конфігурацію операційної системи, під керуванням якої працює веб-додаток. Кожна операційна система дозволяє створювати контрольні аркуші безпеки (security checklist). Ці установки повинні бути узгоджені з операційними системами вендорів. Важливо також, щоб новий додаток вчасно вносився в security cheklist, а не відключав його.

Третій рівень орієнтовано вже на мережу. Необхідно оснастити датчиками атаки мережного обладнання та програмного забезпечення провайдера, що забезпечує хостинг. Головне, щоб сигнал, що надійшов про небезпеку, був правильно оброблений і нейтралізований.

Четвертий рівень безпеки – встановлення програмного забезпечення на рівні хостингу. Це значно складніше завдання. По-перше, тут можна зіткнутися із запереченнями самої хостинг-компанії. А по-друге, таке програмне забезпечення набагато складніше простих датчиків. З цієї причини і рівень безпеки вищий.

Рівень 5 має два підрівні - A та B. Рівень 5A - це встановлення спеціального програмного забезпечення, що виконує роль прошарку між операційною системою веб-сервера та всіма додатками. Такий буфер дозволяє запобігти атакі хакерів на вразливі програми, які беруть під контроль всю операційну систему під час виконання. Це не найдешевший варіант, тому що, як і на попередньому рівні, необхідно забезпечити підтримку програмного забезпечення, яким оперують веб-сервери.

Рівень 5B є установкою орієнтованих на конкретні програми firewall або проксі-серверів. Вони орієнтовані на HTTP-протокол і дозволяють запобігти атакам, перш ніж потенційні зловмисники зможуть дістатися до

запуску додатків, встановлених на веб-сервері. Однак, проксі-сервер – це суттєве обмеження у роботі.

Шостий рівень – своєрідна вершина безпеки. Тут допускається використання лише довірчих операційних систем та працюючих під їх керуванням додатків. Іншими словами, всі функціонуючі додатки та операційні системи повинні бути максимально адаптовані, або розроблені спеціально для специфічних потреб компанії. Це найдорожчий, але й найефективніший спосіб захисту. До того ж вимагає спеціальної підготовки від адміністратора мережі, а часто й від користувачів, що також тягне за собою додаткові витрати.

1.4 Висновки до першого розділу

Результати дослідження показують, що аналіз вихідного коду веб-застосунків дозволяє знайти велику кількість вразливостей різного ступеня ризику і в процесі розробки веб-додатків значно підвищити захищеність кінцевого продукту. У цьому незалежно від стадії розробки доцільно застосовувати автоматизовані засоби, оскільки швидкість роботи аналізатора перевищує можливості ручного аналізу.

Також важливо відзначити, що після виявлення вразливостей у коді веб-програми може знадобитися значний час для їх усунення. У цей час продуктивні системи залишаються вразливими. Тому для їх ефективного захисту важливо не тільки регулярно проводити аналіз захищеності веб-додатків, у тому числі з використанням автоматизованих засобів, але й використовувати превентивні засоби захисту, такі як міжмережевий екран рівня додатків, для виявлення та запобігання атакам на веб-ресурси .

2 ОГЛЯД ЗАСОБІВ ЗАХИСТУ ВЕБ-ДОДАТКІВ. СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

2.1 Методи пошуку вразливостей. Перспективні методи захисту

Як вже зазначалося у першому розділі, сьогодні веб-додатки стають найпопулярнішим інструментом, що пропонує користувачам набір різних послуг. Однак попередні дослідження показали, що багато веб-додатків можуть піддаватися різноманітним вразливостям.

Наступний невичерпний список функцій слід переглядати під час тестування безпеки веб-додатків. Неправильна реалізація кожного з них може призвести до уразливостей [11]:

- Конфігурація програми та сервера. Можливі дефекти пов'язані зі змінами шифрування / криптографії, конфігураціями веб-сервера тощо;

- Перевірка введення та обробка помилок. SQL-ін'єкції, міжсайтові сценарії (XSS) та інші поширені вразливості ін'єкцій є результатом поганої обробки введення та виведення;

- Аутентифікація та управління сеансами. Уразливості, які можуть призвести до видачі себе за іншу особу. Також слід враховувати повноваження та захист;

- Авторизація. Тестування здатності програми захищатися від вертикального та горизонтального підвищення привілеїв.

- Бізнес-логіка. Важлива для більшості програм, які забезпечують бізнес-функції.

- Клієнтська логіка. У сучасних веб-додатках з великою кількістю JavaScript, крім веб-сторінок, що використовують інші типи клієнтських технологій (наприклад, Silverlight, Flash, Java-аплети), цей тип функцій стає все більш поширеним.

Отже, тестування веб-безпеки спрямоване на виявлення вразливостей у веб-додатках та їх конфігураціях. Первинною метою є прикладний рівень (тобто те, що виконується за протоколом HTTP). Тестування безпеки веб-програми часто включає відправку різних типів вхідних даних, щоб спровокувати помилки і змусити систему поводитися несподіваним чином. Ці так звані негативні тести перевіряють, чи робить система те, для чого вона не призначена [11].

Також важливо розуміти, що тестування веб-безпеки - це не лише тестування функцій безпеки (наприклад, аутентифікації та авторизації), які можуть бути реалізовані у додатку. Не менш важливо перевірити, що інші функції реалізовані безпечним способом (наприклад, бізнес-логіка та використання правильної перевірки введення та кодування виведення). Ціль полягає в тому, щоб забезпечити безпеку функцій, представлених у веб-додатку.

Згідно з дослідженнями OWASP [8] найбільш ефективним способом виявлення вразливостей web-додатків є експертний аналіз вихідних кодів web-додатку (code review). Цей спосіб дуже трудомісткий, вимагає високої кваліфікації експерта та не захищений від помилок експерта.

Тому активно також розвиваються методи автоматичного виявлення вразливостей web-додатків.

Методи автоматичного виявлення вразливостей web-сайтів можна розділити на дві основні групи:

- Методи, що аналізують роботу розгорнутого на стенді web-сайту без звернення до вихідних кодів web-сайтів;

- Методи, що аналізують вихідні коди web-сайтів та конфігураційні налаштування. Перша група методів розглядає веб-сайти з погляду зовнішнього користувача, тобто потенційного зловмисника

Згідно з класифікацією, наведеною в [8,11], види тестів безпеки веб-додатків діляться на:

- Тест динамічної безпеки програм (DAST). Цей автоматичний тест безпеки програм найкраще підходить для внутрішніх програм з низьким рівнем ризику, які повинні відповідати нормативним вимогам до безпеки. Для додатків із середнім ступенем ризику та критичних додатків, що зазнають незначних змін, найкращим рішенням є поєднання DAST із деяким ручним тестуванням веб-безпеки на загальні вразливості.

- Статичний тест безпеки додатків (SAST). Цей підхід до забезпечення безпеки програм пропонує автоматизовані та ручні методи тестування. Він найкраще підходить для виявлення помилок без необхідності запускати програми у виробничому середовищі. Це також дозволяє розробникам сканувати вихідний код та систематично знаходити та усувати уразливості безпеки програмного забезпечення.

- Тест на проникнення. Цей ручний тест безпеки додатків найкраще підходить для критично важливих додатків, особливо тих, які зазнають серйозних змін. Оцінка включає бізнес-логіку та тестування зловмисників для виявлення складних сценаріїв атак.

- Самозахист програм під час виконання (RASP). Цей підхід до безпеки програм, що розвивається, включає в себе ряд технологічних прийомів для інструментарію програми, щоб можна було відстежувати атаки в міру їх виконання і, в ідеалі, блокувати в режимі реального часу.

Детально методи автоматичного тестування будуть розглянуті далі у рамках магістерської дисертації.

Виявивши основну причину вразливостей, на ранніх етапах тестування можна реалізувати заходи для пом'якшення наслідків, щоб запобігти будь-яким проблемам. Крім того, знання того, як ці атаки працюють, можна використовувати під час тестування безпеки веб-додатків.

Визнання впливу атаки також є ключем до управління ризиками, оскільки результати успішної атаки можна використовувати для оцінки загальної серйозності вразливості. Якщо проблеми виявляються під час

перевірки безпеки, визначення їхньої серйозності дозволяє організації ефективно розставити пріоритети у зусиллях виправлення.

Перед виявленням проблеми оцінка потенційного впливу на кожну програму в бібліотеці додатків може полегшити пріоритет тестування безпеки додатків. Маючи встановлений список висококласних програм, тестування безпеки web може бути заплановане так, щоб в першу чергу націлити критично важливі програми компанії з більш цілеспрямованим тестуванням, щоб знизити ризики [12].

2.2 Застосування сканерів та їх можливості

Для пошуку вразливостей використовують спеціальні сканери вразливостей. До основних завдань сканерів уразливостей відносяться [13]:

- ідентифікація та аналіз загроз;

- інвентаризація ресурсів (операційна система, програмне забезпечення, мережеві пристрої);

- формування звітів з описом уразливостей та варіанти виправлення уразливостей системи.

Існує два основних механізми роботи сканерів уразливостей [13]:

- сканування - пасивний аналіз, що являє собою збір інформації про порти, операційні системи, версії програмного забезпечення, отримані дані порівнюються з правилами визначення інформації про порти, програмне забезпечення та інші компоненти системи та формується висновок про наявність чи відсутність уразливостей;

- зондування - активний аналіз, що є імітацією атаки вразливість, яка перевіряється.

Існують різні засоби пошуку уразливостей для різних етапів перевірки систем на наявність уразливостей, від сканерів портів до комплексних систем, які складаються із сканерів портів, засобів пошуку експлойтів для вразливостей [13].

Прикладами засобів пошуку уразливостей можуть бути:

- програма для дослідження мережі Nmap – дозволяє визначати хости в мережі, встановлені служби та їх версію, операційні системи та версії, які вони використовують, пакетні фільтри/міжмережові екрани [13];

- комплекс засобів перевірки веб-додатків Burp Suite - має власний проксі-сервер, сканер уразливостей, засіб автоматизації атак;

- Metasploit Framework – інструмент для перевірки та експлуатації вразливостей.

Засоби пошуку та експлуатації вразливостей можуть бути зібрані у спеціальних операційних системах, призначених для аналізу захищеності та проведення тестувань комп'ютерних систем на проникнення.

Існують такі операційні системи для проведення тестування комп'ютерних систем на проникнення:

- ОС Kali Linux;

- ОС BlackArch Linux;

- ОС Parrot Security OS;

- ОС BlackBox.

Розглянемо приклади існуючих сканерів вразливостей (таблиця 2.1).

Таблиця 2.1 – Існуючі сканери вразливостей та їх порівняння

Назва	Defect Tracking Integration	IAST Module Hybrid Analysis	SAST Module Hybrid Analysis	Flash Scanner	CGI Scanner	Enterprise Console Management Features	Demo
Rapid appspider	+	-	-	+	-	+/-	+
Portswigger Burp Suite	+/-	+	-	+	+	+/-	+
Fortify WebInspect	+	+	+	+	+	+	+

Продовження таблиці 2.1

IBM Security AppScan	+	+	+	+	+	+	+
Accunetix Vulnerability Scanner	+	+	-	-	+	+	+
Netsparker Web Application Security Scanner	+	-	-	-	+	+	+
Janusec WebCruiser	-	-	-	-	+	-	+

Розглянемо коротко кожен із запропонованих сканерів.

1) Продукт компанії Rapid7 – це функціональний засіб тестування веб- та мобільних додатків. Крім виявлення вразливостей, Appspider вміє надавати конкретні рекомендації щодо їх усунення та розставляти пріоритети.

Є 95 видів атак, завдяки чому програми можна перевірити на стійкість до всіх сучасних поширених загроз. До того ж тут є можливість запускати певні атаки окремо, щоб перевірити актуальність захисту від них.

Інструмент чудово працює з усіма сучасними технологіями. Є підтримка Ajax, JSON, GWT та інших стандартів, різноманітних форматів та протоколів, що використовуються у сучасних веб-додатках та браузерах. Також є перевірка на відповідність сучасним стандартам безпеки.

Rapid7 може виводити інтерактивні звіти, які подаються у вигляді веб-сторінок.

Їхня головна зручність — це можливість заглибитись у конкретну вразливість та деталізувати її найдрібніші подробиці для вирішення проблеми. Інструмент вміє інтегруватися з наявними засобами безпеки

(наприклад, Web Application Firewall), що істотно економить час і ресурси. Спробувати інструмент можна безкоштовно протягом обмеженого часу.

Інтерфейс сканера наведений на рисунку 2.1.

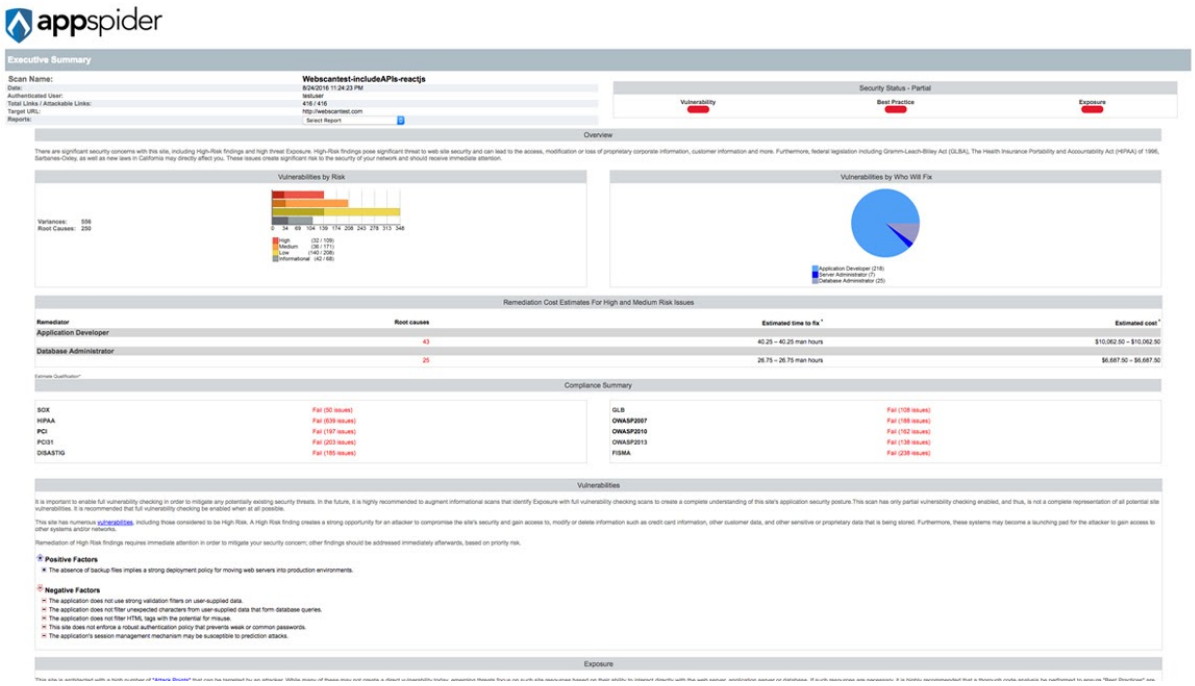

Рисунок 2.1 – Інтерфейс сканера Rapid appspider

2) Portswigger Burp Suite. Цей сканер дозволяє тестувати безпеку веб-додатків як в ручному, так і в автоматичному режимі. Але головний упор робиться на ручні перевірки. Для цього в Burp Suite є низка інструментів. Наприклад, Intruder дозволяє виявляти незвичайні вразливості, а Repeater використовується для маніпуляцій та повторного надсилання індивідуальних запитів. Інтерфейс Burp Suite інтуїтивно зрозумілий, можливо швидко розібратися в програмі.

Однією з переваг інструменту є наявність безлічі сторонніх плагінів і доповнень, які серйозно розширюють його базову функціональність. Багато хто з них розроблений самими програмістами, які використовували Burp Suite у своїй роботі і точно знали, чого саме не вистачає оригінального продукту. Якщо потрібного плагіна знайти не вдалося, його можна написати самостійно.

Інтерфейс програми наведений на рисунку 2.2.

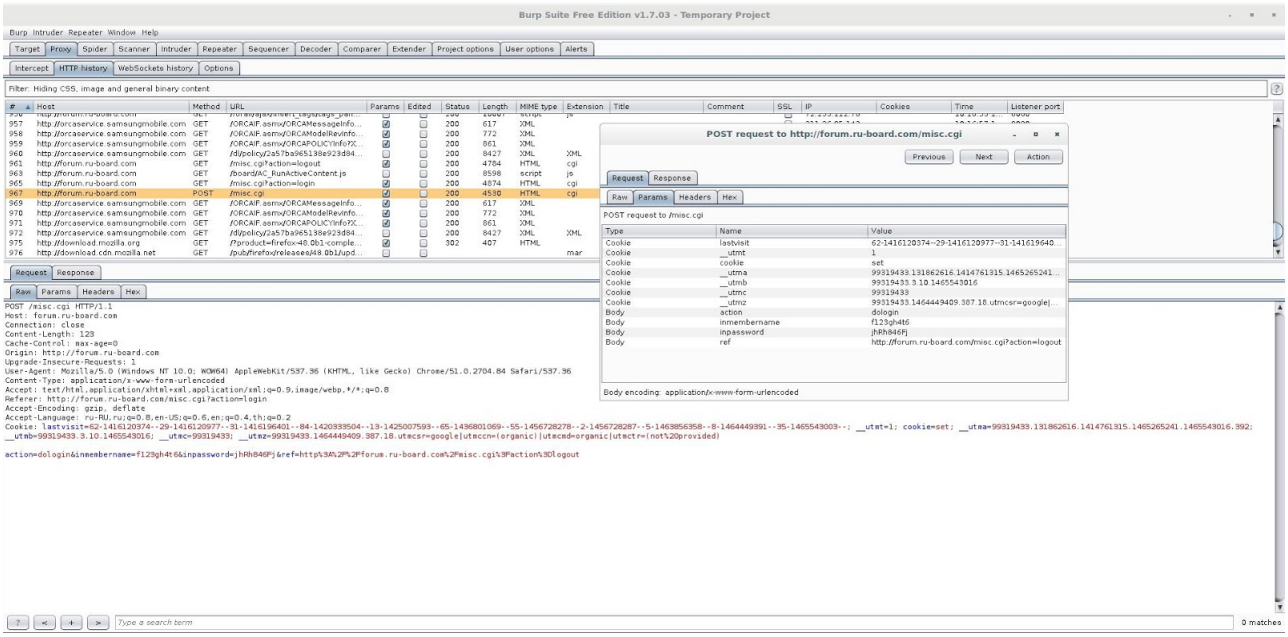

Рисунок 2.2 – Інтерфейс Portswigger Burp Suite

3) Fortify WebInspect. Fortify WebInspect – це один із найфункціональніших інструментів. Він може поставлятися як ліцензоване ПЗ або використовуватися за моделлю SaaS (програмне забезпечення як послуга), а також працювати певний час як демонстраційна версія.

Засіб вміє імітувати реальні атаки та техніки злому, які найчастіше застосовують кіберзлочинці. Сканер підтримує всі сучасні технології, що дозволяє працювати з додатками без огляду на його архітектуру. Серед найпопулярніших — Adobe Flash та JavaScript/Ajax, які на сьогоднішній день використовуються при створенні веб-застосунків дуже часто.

До переваг цього продукту також належать простота установки, налаштування та можливість масштабування. Після закінчення тестів Fortify WebInspect видає найдокладніші звіти, які будуть корисні та зрозумілі як менеджменту компанії, так і розробникам. У них показується статистика знайдених уразливостей, їхня пріоритетність (на що потрібно звернути увагу перш за все), демонструються докладні відомості про кожну проблему. У програмі є набір готових шаблонів для звітів, але можна також створювати свої власні.

Інтерфейс наведений на рисунку 2.3.

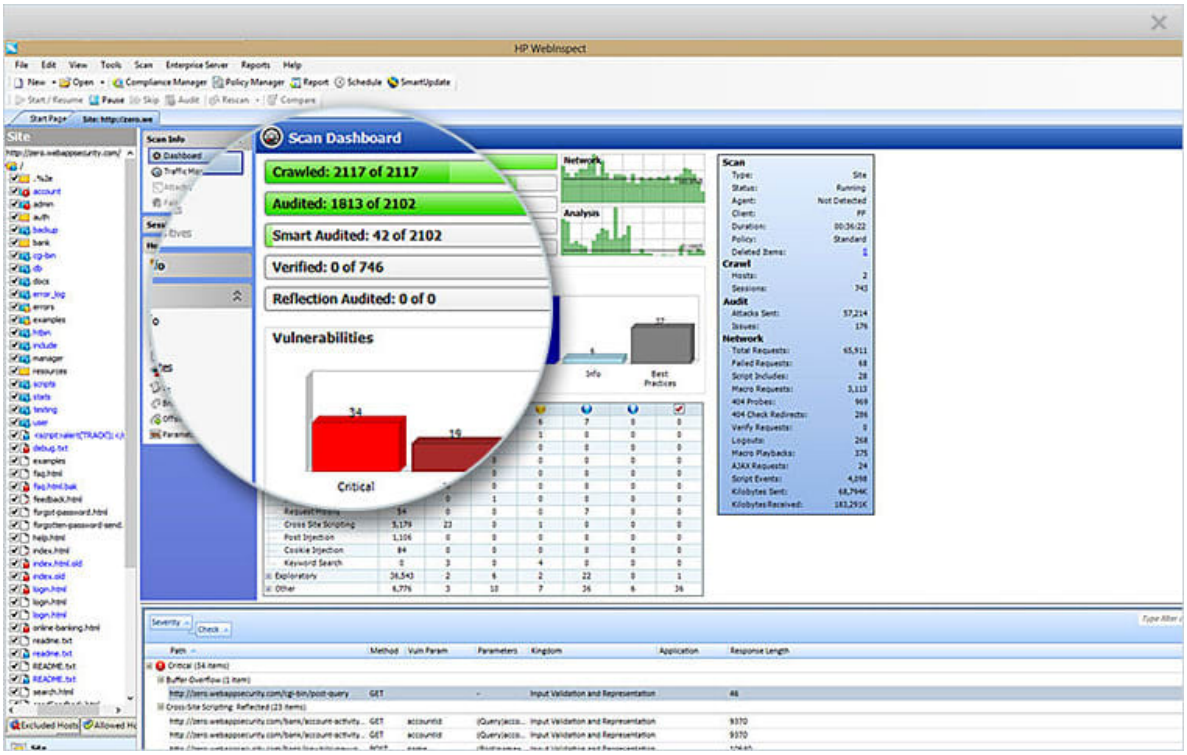

Рисунок 2.3 – Інтерфейс Fortify WebInspect

4) IBM Security AppScan. В арсеналі Security AppScan є різноманітні інструменти для проведення статичних та динамічних тестів, а також перевірки Open Source компонентів.

Програма підтримує більшість сучасних протоколів, стандартів та архітектур, у тому числі JavaScript/Ajax та Adobe Flash. Хоча упор для IBM Security AppScan зроблений на автоматичних перевірках, є можливість проводити і ручні тести додатків. Алгоритми перевірки для максимальної схожості з реальними атаками використовують адаптивні процедури, що імітують людську поведінку.

Виявлені проблеми подаються у вигляді зручних звітів. У базі програми є понад 40 шаблонів звітів про відповідність різноманітним стандартам: ISO 27001, ISO 27002, Basel II, т.д.

Для кожної виявленої вразливості наводиться докладне пояснення та рекомендації щодо оперативного усунення проблеми.

У цих порадах використовуються підготовлені етапи роботи, що включають приклади коду та список завдань, які необхідно виконати першочергово. Продукт постачається в кількох редакціях, перед покупкою можна спробувати безкоштовну версію.

Інтерфейс наведений на рисунку 2.4.

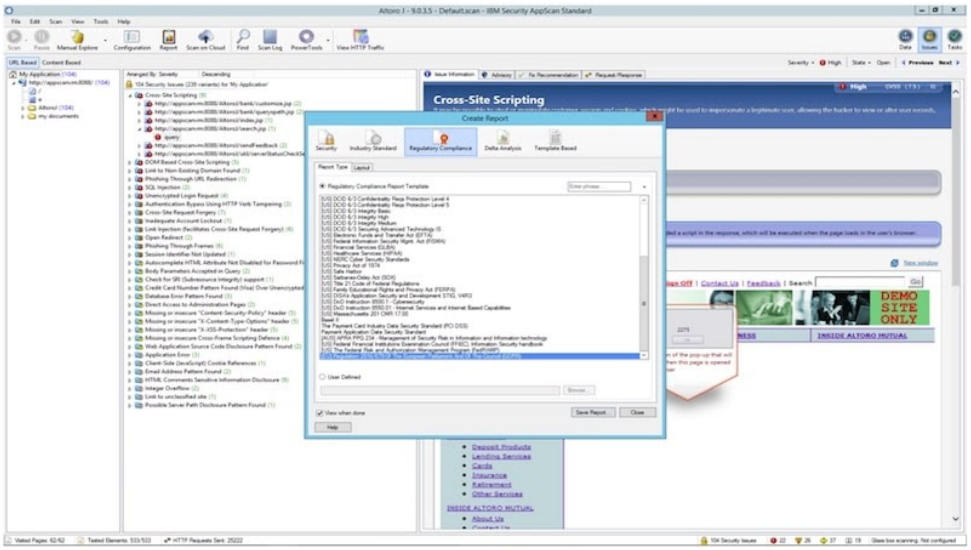

Рисунок 2.4 – Інтерфейс IBM Security AppScan

5) Accunetix Vulnerability Scanner. Захисні продукти Acunetix користуються заслуженою популярністю серед таких клієнтів, як, Visa або American Express. Серед цих продуктів і сканер захищеності веб-застосунків Acunetix Vulnerability Scanner. Цей інструмент має великий набір функцій для забезпечення автоматичного контролю безпеки програм.

Він виявляє всі поширені типи вразливостей, серед яких SQL-ін'єкції, виконання шкідливих скриптів та кодів. Наприклад, для популярної платформи WordPress цей продукт може виявити до 1200 відомих уразливостей. При цьому він може працювати у багатопотоковому режимі, що дозволяє перевіряти тисячі об'єктів різних платформ без перерви.

Усі виявлені проблеми відображаються у зручних звітах. Вони підходять як для фахівців, які безпосередньо усуватимуть проблеми, так і для керівників, яким потрібно бути в курсі того, що відбувається, і розуміти

загальну картину. Підсумкові звіти можуть бути скомпоновані у загальному файлі. Ці результати можуть бути звірені з аналогічними даними після минулих перевірок, щоб визначити, які уразливості усунуті, а які ще залишилися.

Пробна версія програми (з деякими обмеженнями) доступна протягом 14 днів. Також сервіс має хмарний сканер, який передбачає деяку кількість безкоштовних перевірок.

Інтерфейс програми наведений на рисунку 2.5.

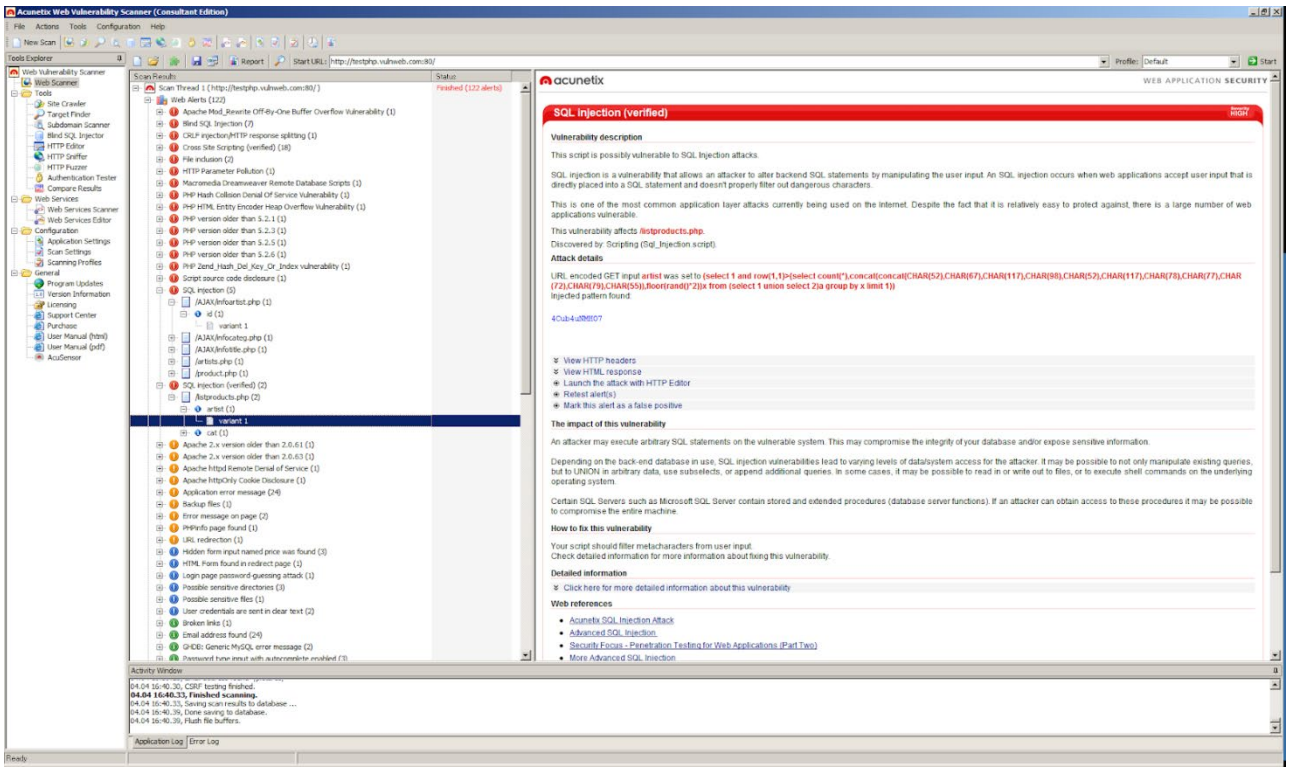

Рисунок 2.5 - Accunetix Vulnerability Scanner

6) Netsparker Web Application Security Scanner. Цей повністю автоматичний сканер відрізняється високим рівнем виявлення вразливостей та мінімальною кількістю помилкових спрацьовувань. Такий результат досягається завдяки фірмовому інструменту Proof-Based Scanning, який не лише сигналізує про загрозу, а й одразу ж надає докази, що це не хибне спрацювання.

Таким чином економиться багато часу та ресурсів, адже виявлені загрози не потрібно повторно перевіряти вручну, а можна одразу приступати до їх усунення.

Netsparker Web Application Security Scanner вміє аналізувати веб-програми та сервіси на всіх поширених платформах, серед яких Java Script, HTML 5, .NET та багато інших. Перевірка проводиться у всіх поширених типах атак.

Інструмент може одночасно працювати з сотнями та тисячами ресурсів, при цьому легко інтегрується у вже існуючі системи безпеки. Продукт поставляється в десктопному, корпоративному та хмарному варіанті. Також є пробна версія.

Інтерфейс наведений на рисунку 2.6.

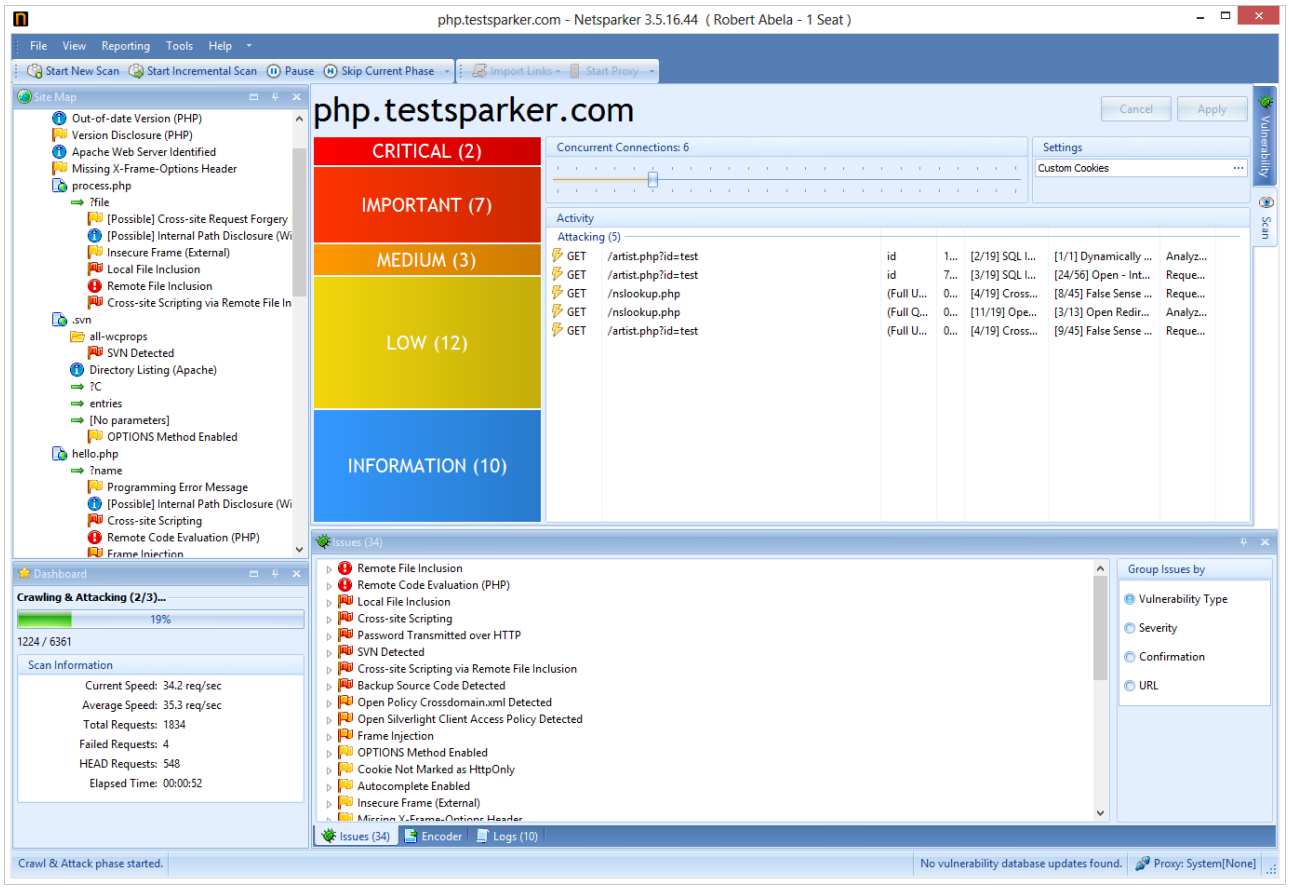

Рисунок 2.6 – Інтерфейс Netsparker Web Application Security Scanner

7) Janusec WebCruiser. Сканер заточений головним чином на проведення SQL-ін'єкцій на різних платформах, наприклад, SQL Server, Oracle і Access. У ньому також є інструменти для роботи з cookie, міжсайтовим криптингом, PHP-ін'єкціями та іншими найпоширенішими загрозами.

Інтерфейс наведений на рисунку 2.7.

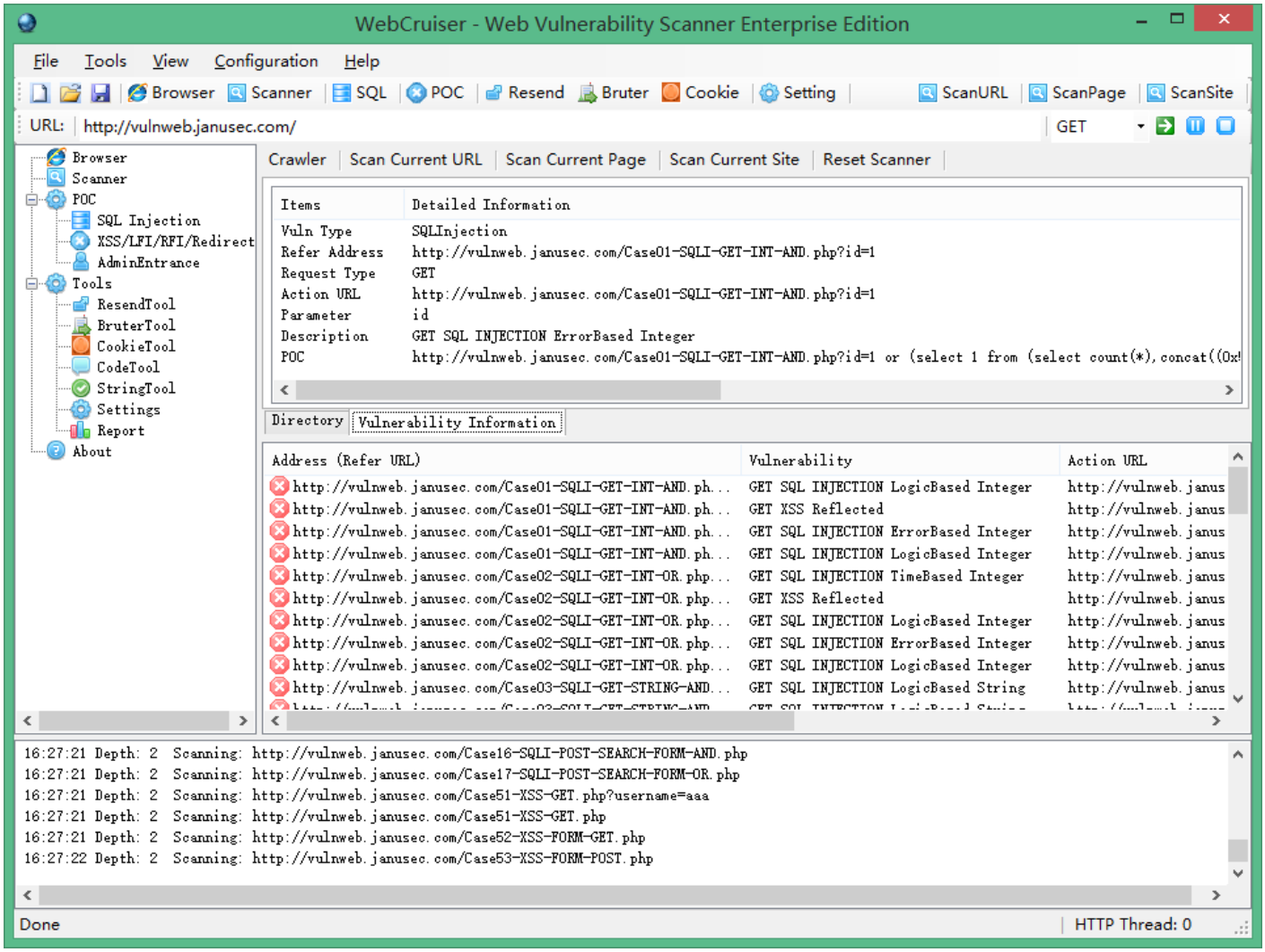

Рисунок 2.7 – Інтерфейс Janusec WebCruiser

8) Nessus. Це добре відомий та популярний сканер уразливостей, який надається безкоштовно для особистого некомерційного використання.

Його було вперше випущено в 1998 році Ренурдом Дерасоном і в даний час опубліковано Tenable Network Security. Існує також додатковий проект Nessus 2 під назвою OpenVAS, який публікується за ліцензією GPL.

Використовуючи велику кількість перевірок уразливостей, які називаються плагінами в Nessus, можна ідентифікувати велику кількість добре відомих уразливостей. Metasploit приймає файли результатів сканування вразливостей як від Nessus, і від OpenVAS у форматі файлу nbe .

Інтерфейс сканера наведений на рисунку 2.8.

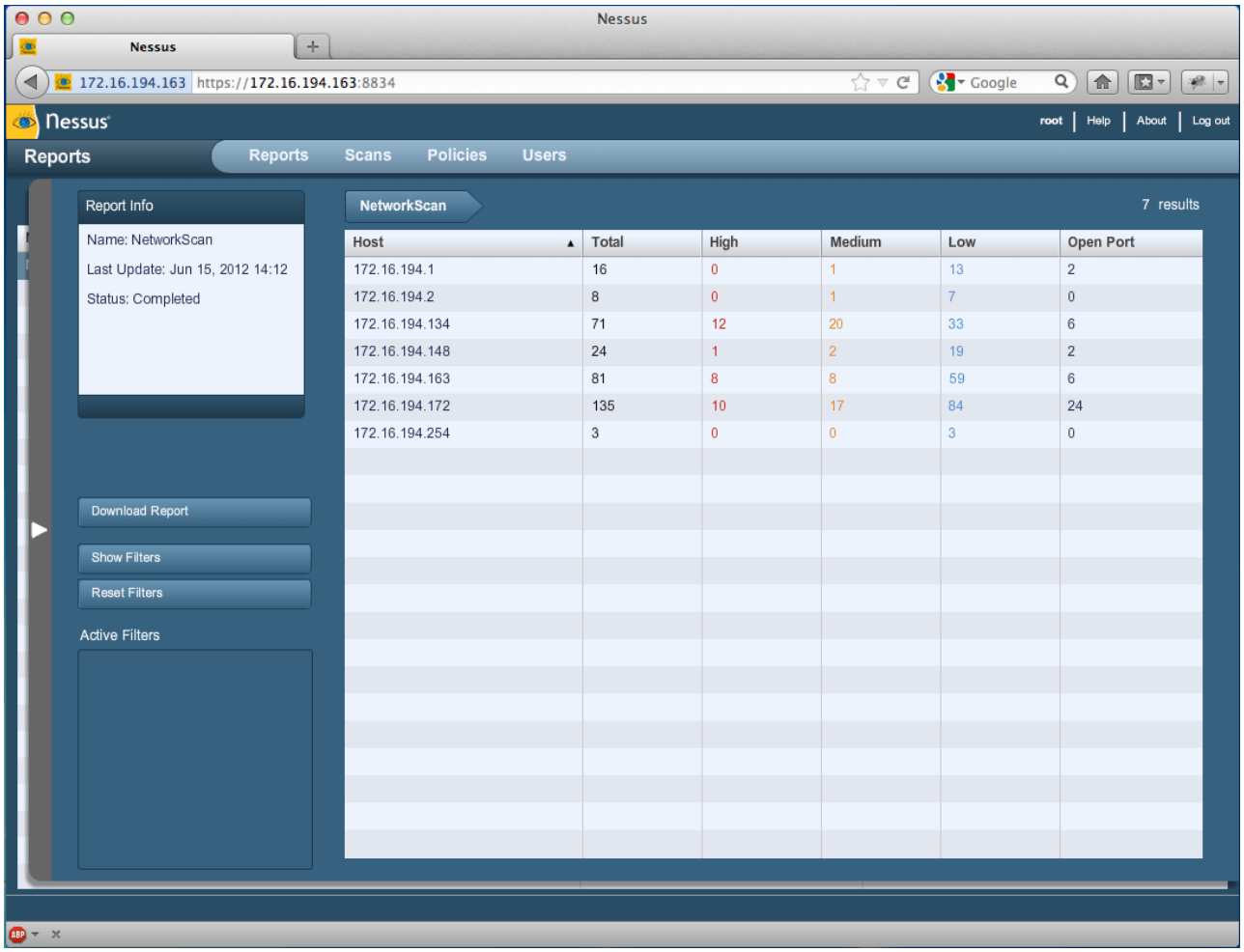

Рисунок 2.8 - Інтерфейс сканера Nessus

Nessus може сканувати наступні вразливості:

- Вразливості, які можуть дозволити неавторизований контроль або доступ до конфіденційних даних у системі;

- Неправильна конфігурація (наприклад, відкритий поштовий ретранслятор):

- Відмова в обслуговуванні (DOS) вразливості;

- Стандартні паролі, кілька спільних паролів та порожні / відсутні паролі для деяких системних облікових записів

Збої в програмному забезпеченні, відсутні виправлення, шкідливе програмне забезпечення та помилки неправильної конфігурації в широкому спектрі операційних систем, пристроїв та програм вирішуються за допомогою Nessus.

Сервер Nessus в даний час доступний для:

- Unix;

- Linux;

- FreeBSD.

Також сканер доступний для:

- Операційних систем на основі Unix;

- Операційних системи на базі Windows;

Істотні можливості Nessus включають:

- Заплановані аудити безпеки;

- Виявлення «дірок» у безпеці на локальних чи віддалених хостах;

- Імітація атак виявлення вразливостей;

- Виявлення відсутніх оновлень безпеки та патчів;

Nessus Professional виконує сканування внутрішньої мережі відповідно до вимог PCI DSS 11.2.1.

Приклад результатів сканування наведений на рисунку 2.9.

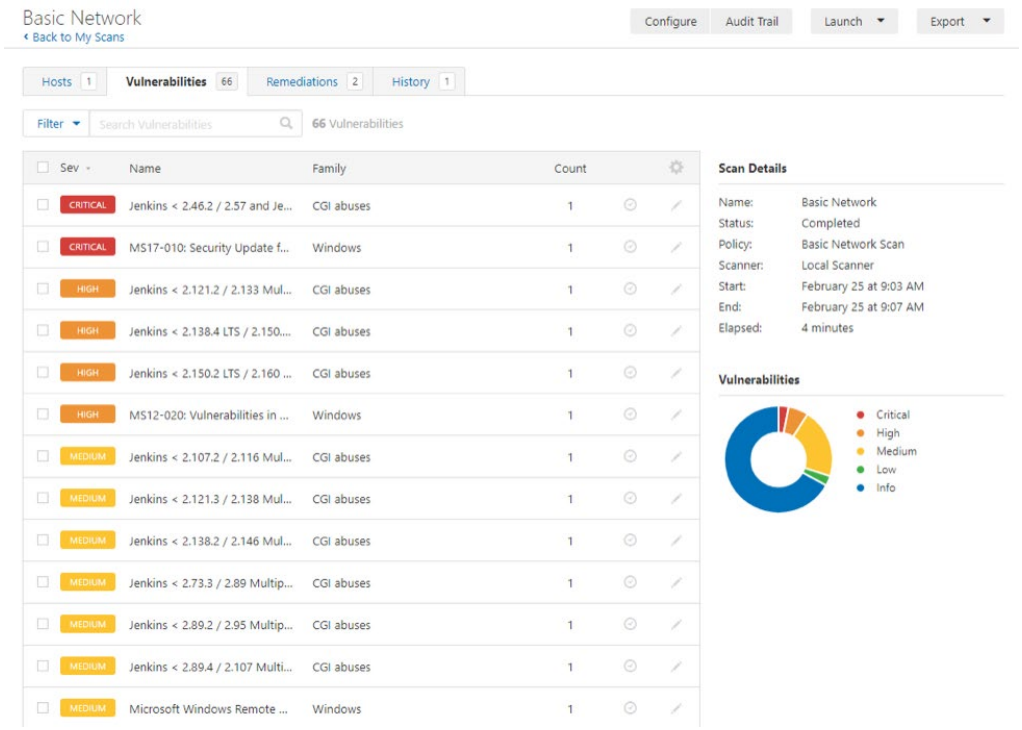

Рисунок 2.9 – Приклад результатів сканування Nessus

Таким чином, існують наступні варіанти пошуку вразливостей - використання комплексних комерційних рішень, таких як Nessus, або використання засобів, що входять до спеціальних ОС, таких як Kali Linux та ін.

Використання спеціальних ПЗ для пошуку вразливостей та проведення тестувань на проникнення передбачає дотримання певного алгоритму застосування вбудованих в ПЗ засобів з метою підвищення ефективності пошуку вразливостей.

2.3 Системи автоматизированого тестування як ефективні засоби захисту веб-додатків

При ручному тестуванні веб-додатків витрачається багато людських ресурсі для проведення повного циклу тестування безпеки під час виконання наступних етапів:

- Збір інформації;
- Ручне сканування вразливостей веб-додатків;

- Виявлення та перевірка вразливості;

- Складання звітів;

- Виправлення [13].

У будь-якій організації ці етапи зазвичай пов'язані з залученням безлічі людей: розробників, аналітиків якості, менеджерів проектів, адміністраторів мережі, розробників веб-додатків, менеджерів інформаційної безпеки, аудиторів та керівників. Навіть сторонні постачальники залучаються до проектів оцінки веб-безпеки. З такою кількістю співробітників, які працюють заради спільної мети, необхідно максимально автоматизувати пошук вразливостей веб-додатків.

Є певні переваги автоматизованого тестування веб-додатків. По-перше, є ризик дублювання зусиль під час надлишкових тестів. Коли є безліч складних веб-додатків, як більшість сучасних онлайн-компаній, це може призвести до значного обсягу непотрібної роботи [13].

Ще одна проблема, полягає в тому, що не вистачає знань або часу для постійного ручного тестування вразливостей усіх веб-додатків. Якщо тестування безпеки веб-застосунків не автоматизовано з використанням перевіреного автоматизованої системи безпеки веб-застосунків, які можуть перевірити тисячі потенційних недоліків безпеки, при цьому деякі, якщо не всі серйозні вразливості веб-додатків можуть бути втрачені.

В якості прикладу можна розглянути індивідуалізовану веб-систему планування ресурсів підприємства (ERP). Така система буде мати сотні видимих точок входу або поверхонь для атак, а також безліч інших прихованих вразливостей, які необхідно перевірити на наявність вразливостей веб-додатків, таких як SQL-ін'єкції та міжсайтові сценарії.

Нехай ERP має 200 точок входу, які необхідно перевірити на предмет 100 різних варіантів вразливості веб-додатків. Це означає, що тестеру на проникнення необхідно запустити щонайменше 20 000 тестів безпеки. Якби кожен тест займав всього 5 хвилин, спеціалісту з веб-безпеки знадобилося б

близько 208 робочих днів, щоб завершити належний аудит безпеки веб-застосунків в ERP-системі.

Система автоматизованого захисту веб-додатків може сканувати набагато більші ERP-системи на предмет набагато більшої кількості варіантів уразливостей веб-додатків за декілька годин. І, на відміну від людини, автоматичний сканер безпеки не забуде просканувати вхідний параметр.

Виконуючи ручний тест безпеки веб-застосунків, виникає обмеження проникнення рядом відомих уразливостей, відомих тестеру на проникнення. З іншого боку, при використанні автоматичного сканера веб-вразливостей, вможливо переконаєтеся, що всі параметри перевіряються на відповідність усім типам варіантів безпеки веб-додатків.

Використовуючи системи автоматизованого захисту, можна гарантувати гарантуєте, що в результатах сканування безпеки веб-додатків не буде повідомлень про помилкові спрацьовування, тому не потрібно виділяти час на перевірку виявлених уразливостей.

Важливість автоматизації тестування вразливостей підкреслюється популярними дослідженнями у сфері інформаційної безпеки. Рік за роком це дослідження вказує на ті самі основні причини інформаційних ризиків, такі як брак ресурсів, недостатня прозорість і непоінформованість керівництва. Кожен із цих елементів може бути вирішений шляхом автоматизації процесів тестування безпеки [13].

Як відмічається в [13], немає ідеального способу перевірити вразливість веб-безпеки. Однак робити це вручну і покладатися тільки на досвід співробітників не може гарантувати повну достовірність отриманих результатів тестування.

Існують різні інструменти автоматичного тестування безпеки веб-додатків, розглянемо список із найкращих інструментів систем тестування веб-додатків станом на 2021 рік згідно з інформацією, представленою в [12,13].

1. WebLOAD. WebLOAD - це ефективний інструмент тестування продуктивності веб-застосунків для веб-додатків, який показує здатність програми обробляти навантаження. Цей інструмент має потужний потенціал створення сценаріїв, який спрощує комплексне тестування. Більш того, за його допомогою можна визначити аспекти продуктивності, вузькі місця та можливі вразливості у додатку. Це відповідний інструмент, який може допомогти у досягненні необхідної ефективності відгуку для веб-додатку. Крім цього, WebLOAD також інтегрований з Selenium, Jenkins та іншими інструментами, щоб підвищити його можливості тестування.

Інтерфейс наведений на рисунку 2.10.

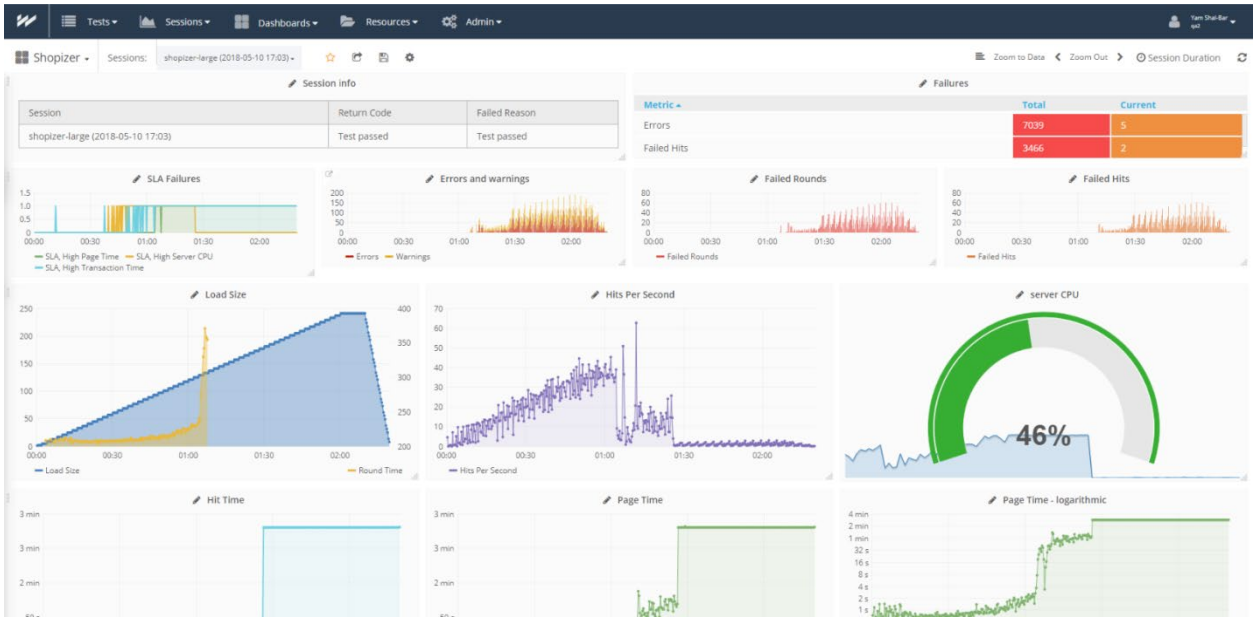

Рисунок 2.10 – Інтерфейс інструменту WebLOAD

2. Acunetix. Acunetix - це автоматизований варіант інструментів тестування безпеки веб-додатків, в який вбудований сканер безпеки, який був розглянутий раніше для виявлення навіть незначних вразливостей більш ніж 4500 веб-додатків.

Інтерфейс наведений на рисунку 2.11.

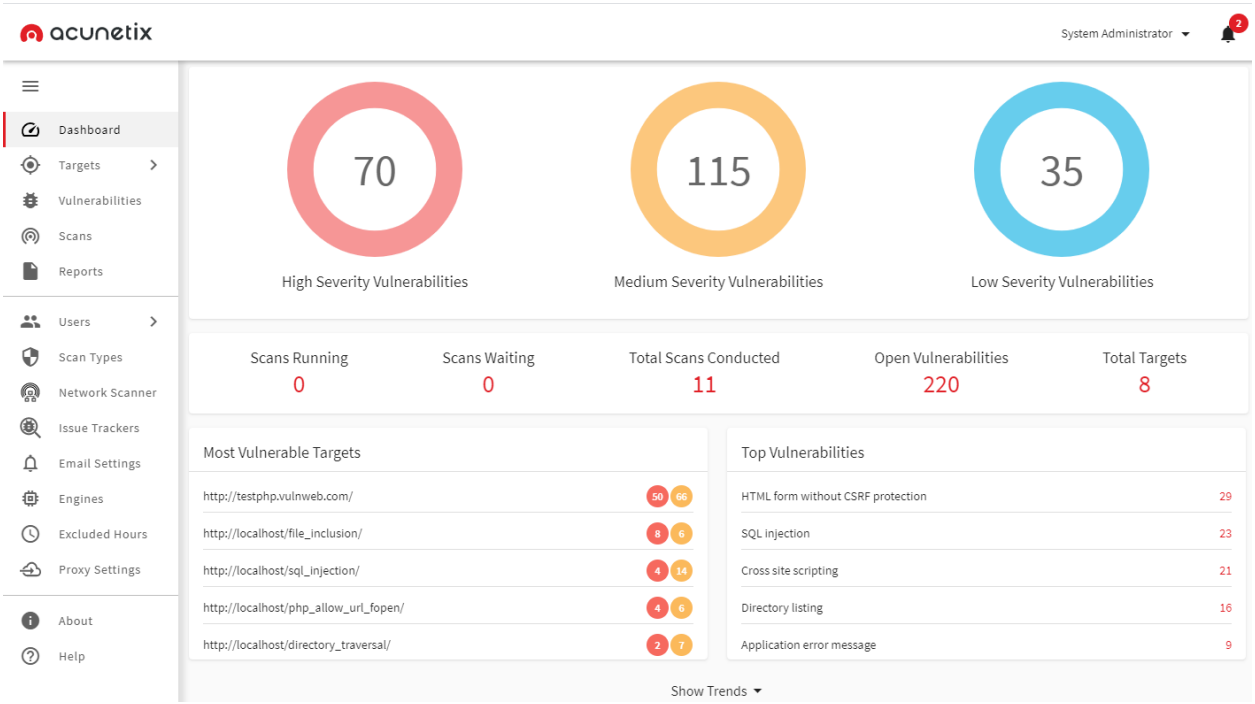

Рисунок 2.11 – Інтерфейс автоматизованої системи Acunetix

3. Netsparker. Інструмент тестування веб-додатків Netsparker відомий своєю точністю. Він також має вбудований сканер безпеки для виявлення дрібних та серйозних уразливостей майже у всіх веб-API, який був розглянутий раніше. Цей інструмент виявляє помилки і перевіряє, чи вони справжні або помилкові. Це допоможе заощадити час у порівнянні з ручною перевіркою та переглядом кожної виявленої несправності після сканування. Можна використовувати інструмент Netsparker як програмне забезпечення Windows або як постачальник послуг онлайн. Функції та ефективність залишаються однаковими в обох версіях.

Інтерфейс наведений на рисунку 2.12.

Рисунок 2.12 – Інтерфейс автоматизованої системи Netsparker

4. Experitest. Experitest дозволяє користувачам тестувати веб-додатки на більш ніж 1000 пристроїв одночасно у хмарі. Існують як інструкції, так і автоматизовані інструменти для перегляду веб-додатків, призначені для тестування веб-застосунків.

З Experitest буде достатньо протестувати програму в будь-якому браузері. Більш того, він інтегрований з Appium та Selenium для досягнення найкращих результатів. Шаблон тестування цього інструменту працює в режимі реального часу та налагоджує його для отримання найкращих результатів. Experitest може провести більше 100 тестів одночасно. На додаток до цього також можна пройти візуальне тестування, щоб визначити ефективність відгуку інтерфейсу користувача при різних дозволах.

Інтерфейс наведений на рисунку 2.13.

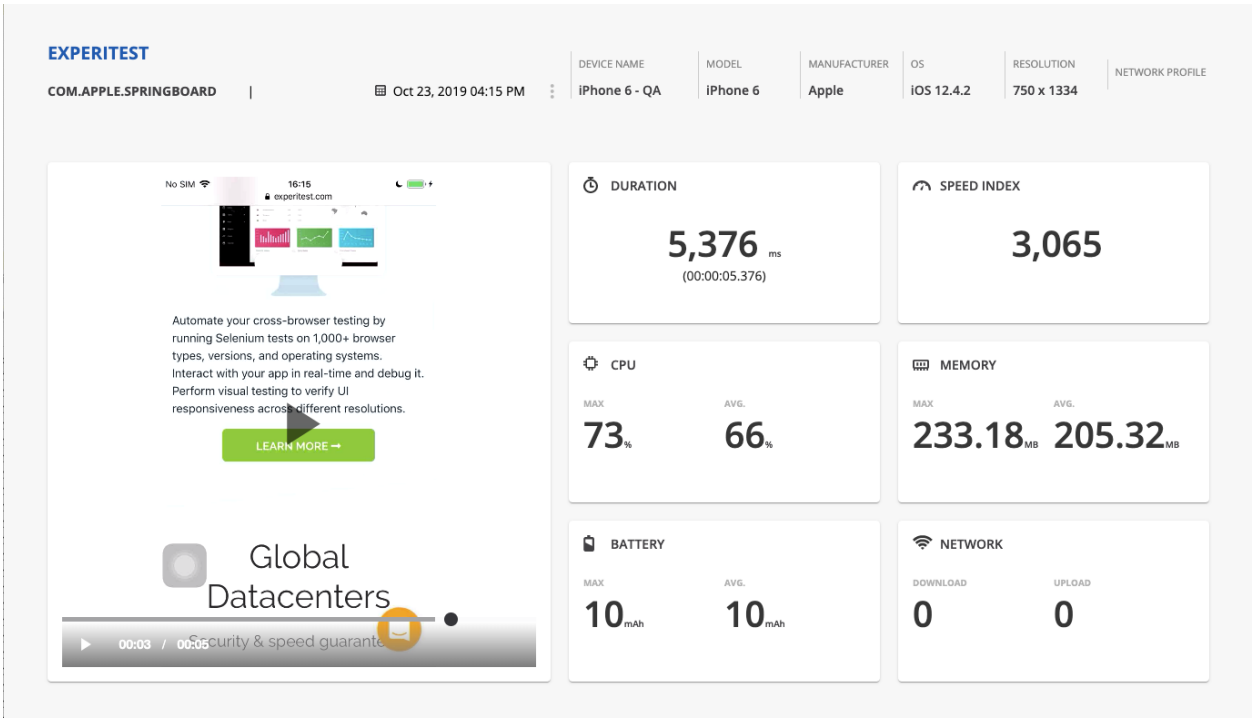

Рисунок 2.13 – Інтерфейс автоматизованої системи Experitest

5. Lambda Test. Інструмент Lambda Test призначений для забезпечення безперебійної роботи всіх елементів веб-застосунків, таких як CSS, HTML5 та інших, на всіх пристроях. Цей інструмент слідує за схемою тестування: ручне, візуальне, а також автоматичне тестування або прогресивні результати. Він використовує хмарну інфраструктуру для одночасного паралельного запуску кількох тестів.

Інтерфейс наведений на рисунку 2.13.

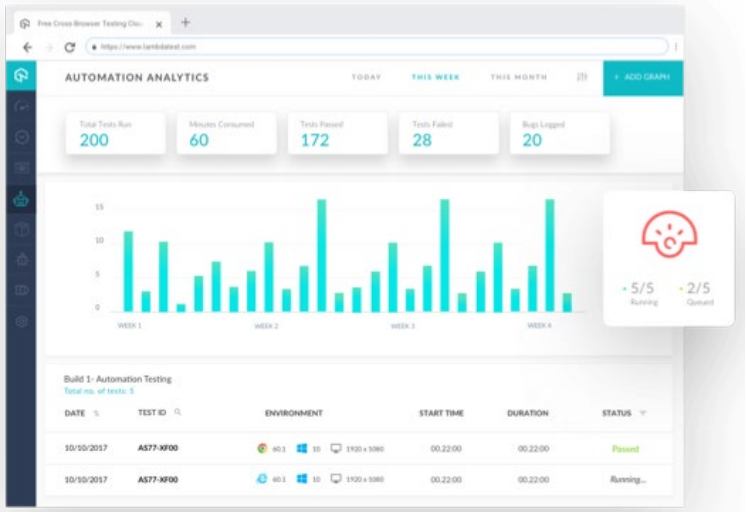

Рисунок 2.13 – Інтерфейс автоматизованої системи Lambda Test

6. Selenium. У списку інструментів автоматизації тестування з відкритим кодом для веб-додатків це один із найбільш широко застосовуваних інструментів тестувальників. Selenium широко відомий як один із найкращих інструментів регресійного тестування, який без проблем працює у різних браузерах та платформах.

Автоматизація тестування – одна з найкращих переваг Selenium, і в нього вбудовано безліч окремих інструментів. Кожен інструмент Selenium призначений для обробки різних аспектів тестування веб-додатків. Selenium IDE, Selenium Web Driver, Selenium RC та Selenium Grid – це чотири компоненти Selenium.

Інтерфейс системи наведений на рисунку 2.14.

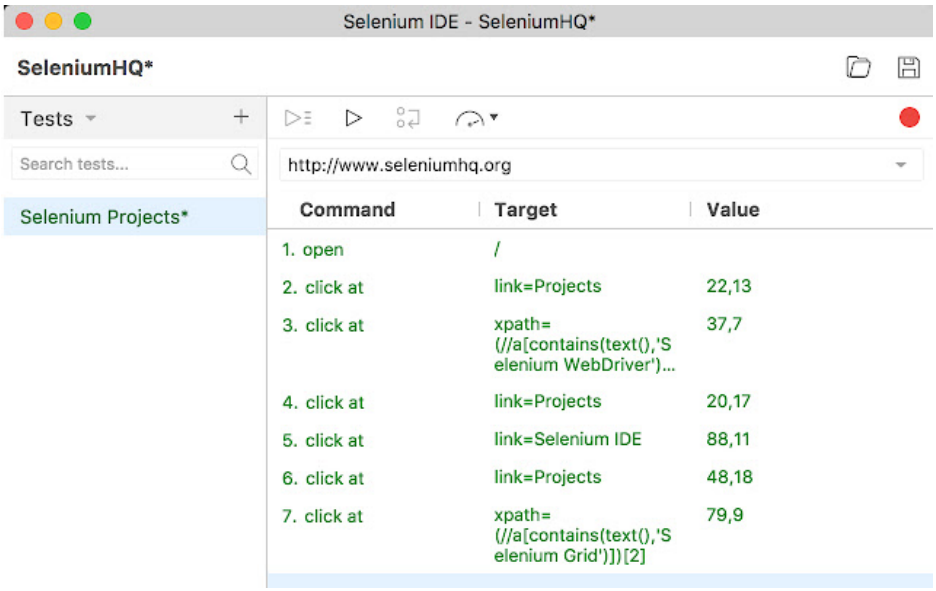

Рисунок 2.14 – Інтерфейс автоматизованої системи Selenium

7. Watir. Watir – це абревіатура від слова «тестування веб-додатків у Ruby». Найкраще в Watir - те, що він тестує веб-програми так само, як і люди. Цей інструмент тестування використовує такі ідеології, як перехід за посиланнями, перевірка текстів, заповнення форм та інші аспекти для тестування веб-додатків у реальному часі для виявлення основних лазівок. Система досить проста у використанні і без проблем працює на різних платформах чи браузерах.

Інтерфейс представлений на рисунку 2.15.

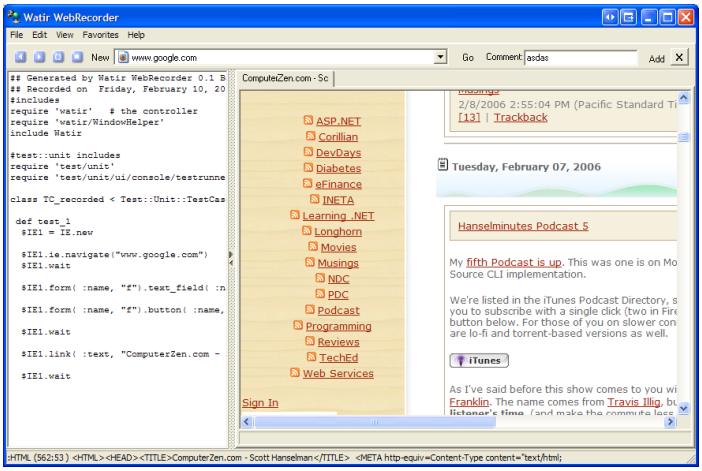

Рисунок 2.15 - Інтерфейс автоматизованої системи Watir

8. Ranorex Studio. Ranorex розроблений для максимального збільшення ресурсів підтримки методів автоматизованого тестування. Це відповідний інструмент для початку наскрізного тестування з використанням симуляторів або реальних пристроїв для більш помітних результатів. Він чудово працює практично з усіма браузерами, такими як Safari, Microsoft Edge, Chrome та іншими.

Інтерфейс представлений на рисунку 2.16.

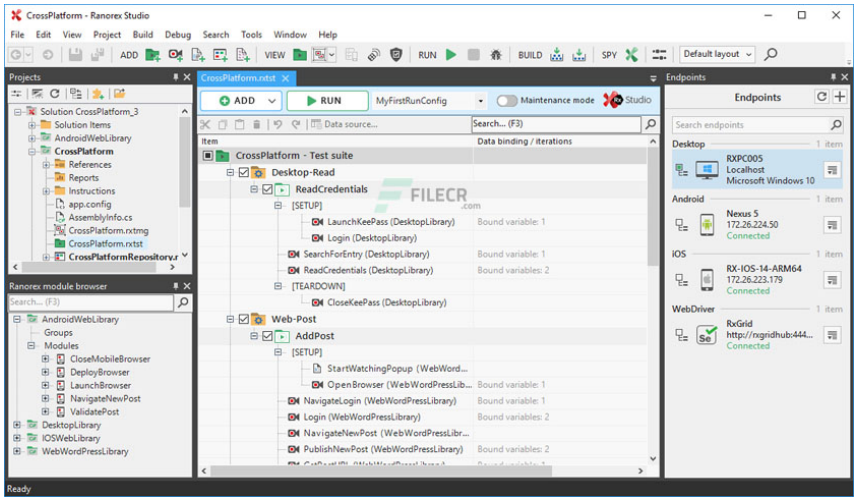

Рисунок 2.16 – Інтерфейс автоматизованої системи Ranorex Studio

9. Serenity. Інструмент тестування Serenity гарантує, що написані тести простіше в обслуговуванні та досить гнучкі для різних програм. Крім того,

Serenity складає докладні звіти про тестування, щоб дати уявлення про продуктивність веб-додатку. Більше того, він також відстежує весь прогрес проекту.

Вище наведено 9 найкращих інструментів для тестування веб-додаткыв, які широко використовуються компаніями. При адекватних результатах робота над модифікацією додатків стане простішою та зручнішою.

2.4 Висновки до другого розділу

Існують різні засоби пошуку уразливостей для різних етапів перевірки систем на наявність уразливостей, від сканерів портів до комплексних систем, які складаються із сканерів портів, засобів пошуку експлойтів для вразливостей.

Для веб-додатків існує окрема ідеологія тестування, і цей процес охоплює безліч аспектів. Під час тестування веб-додатків перевіряються такі аспекти, як безпека веб-сайту, функціональність, зручність використання, доступність, продуктивність та інші. У разі будь-яких проблем чи лазівок інструменти тестування веб-додатків вкажуть розробникам їх усунення в найкоротші терміни.

Актуально використовувати автоматизовані системи тестування веб-додатків. Проте аналіз існуючих рішень показав, що деякі з них є недостатньо ефективними щодо нових виникаючих загроз, є нешвидко дійними та не застосовують сучасних технологій тестування. В межах написання роботи пропонується розробка власної системи автоматизованого тестування з застосуванням нейронних мереж з метою підвищення ефективності виявлення сучасних загроз інформаційної безпеки.

Розробка власної системи автоматизованого тестування веб-додатків буде представлена у наступному розділі магістерської дисертації

3 РОЗРОБКА ТА ДОСЛІДЖЕННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-ДОДАТКІВ

3.1 Методика пошуку потенціальних вразливостей з застосуванням системи автоматизованого тестування

В межах даного розділу магістерської дисертації виконаємо розробку та дослідження системи автоматизованого тестування безпеки веб-додатків.

Спершу виконаємо обґрунтування інструментів проектування та програмних засобів.

Оберемо мову програмування. На сьогоднішній день налічується десятки мов програмування. Проте є найпопулярніші. Розглянемо особливості мов програмування: Python, Java, C, C++, C#, PHP.

Python – це швидка, проста у використанні мова програмування, яка широко використовується для розробки масштабованих веб-додатків: YouTube, Instagram, Pinterest. Python забезпечує відмінну підтримку бібліотек та має велику спільноту розробників.

Переваги:

- простий створення та використання класів та об'єктів;
- велика підтримка бібліотек;
- основна увага приділяється зручності читання коду;
- має можливість масштабувати навіть найскладніші програми;
- ідеально підходить для створення прототипів та тестування ідей;
- має відкритий вихідний код з постійно зростаючою підтримкою;
- забезпечує підтримку для безлічі платформ та систем;
- дуже проста у освоєнні та використанні.

Недоліки:

- не підходить для мобільних обчислень;
- повільна через те, що це інтерпретована мова програмування

- Рівень доступу до бази даних є дещо незрілим.

Java – це популярна мова розробки у великих організаціях, і вона залишається такою протягом десятиліть. Java широко використовується для побудови корпоративних веб-додатків. Java є надзвичайно стабільною. Java також широко використовується в розробці програм для Android.

Переваги:

- велика кількість бібліотек з відкритим вихідним кодом;

- автоматичне виділення пам'яті;

- слідує парадигмі ООП;

- має систему розподілу стека;

- високий рівень незалежності платформи завдяки функції JVM;

- висока безпека за рахунок виключення явного покажчика та включення менеджера безпеки, відповідального визначення доступу класів;

- ідеально підходить для розподілених обчислень;

- пропонує безліч API для виконання різних завдань, таких як підключення до бази даних, мережі, утиліти та синтаксичний аналіз XML;

- підтримка багатопоточності;

Недоліки:

- відсутність обмежень шаблонів створення високоякісних структур даних;

- витратний механізм керування пам'яттю;

- працює повільніше, ніж компілювання мов програмування, такі як C, C++, C#.

Низькорівневі системи, такі як операційні системи, файлові системи тощо, написані на C/C++. C++ також широко використовується конкурентоспроможними програмістами завдяки тому, що є надзвичайно швидкою і стабільною. C++ також надає STL – стандартну бібліотеку шаблонів – пул готових до використання бібліотек для різних структур даних, арифметичних операцій та алгоритмів. Бібліотечна підтримка та швидкість мови роблять C++ популярною мовою.

Переваги:

- безліч компіляторів та бібліотек для роботи з [C++];

- полегшує доступ до заблокованих або прихованих об'єктів за допомогою інших мов програмування [C];

- швидше виконання програм у порівнянні з більшістю інших мов програмування [C/C++];

- формує основу розуміння складніших мов програмування [C/C++];

- мова вибору для розробки додатків з декількома пристроями та декількома платформами [C++];

- процедурно-орієнтована мова з групою функціональних модулів та блоків. Це спрощує налагодження, тестування та обслуговування програм [C];

- програми більш ефективні та прості для розуміння [C/C++];

- велика бібліотека функцій [C++];

- працює близько до апаратного забезпечення системи та, отже, пропонує низький рівень абстракції [C/C++];

- підтримка обробки винятків та навантаження функцій [C++];

- велика різноманітність сфер застосування, таких як ігри, графічні програми та математичне моделювання в реальному часі [C++];

Недоліки:

- складний синтаксис [C/C++];

- не підтримує простір імен [C];

- необхідно створити вручну високорівневі конструкції [C];

- страждає від проблем переповнення буфера та пошкодження пам'яті [C/C++];

- менша стандартна бібліотека [C].

PHP є однією з найпопулярніших мов серверного програмування. Хоча PHP стикається з жорсткою конкуренцією з боку Python і JavaScript, ринок, як і раніше, потребує великої кількості розробників PHP.

Переваги:

- безліч потужних бібліотек;

- низький поріг входу;

- першокласне налагодження за допомогою Xdebug;

- гігантська підтримка спільноти та величезна екосистема;

- безліч засобів автоматизації для тестування та розгортання додатків;

- відсутність дефіциту хороших засобів автоматизації для розгортання та тестування;

- підтримує об'єктно-орієнтовані та функціональні парадигми програмування.

Недоліки:

- розробка програм повністю на PHP відбувається повільніше в порівнянні з використанням інших додаткових мов;

- недостатній базовий рівень безпеки;

- недостатня ефективність обробки помилок;

- вимагає розширення функціональності за рахунок інших мов, зокрема JavaScript.

C# – це універсальна мова програмування, розроблена корпорацією Microsoft. C# широко використовується для програмування серверної та клієнтської частин, побудови ігор (за допомогою фреймворку Unity), побудови програм для мобільних телефонів.

Переваги:

- оскільки не дозволені типи покажчиків, набагато безпечніше, ніж C і C++;

- можливість роботи із загальними базами коду;

- автоматична масштабованість та можливість оновлення;

- компонентно-орієнтована, об'єктно-орієнтована мова програмування;

- повна інтеграція із бібліотекою.NET;

- ідеально підходить для всіх типів розробки Windows;

- багаті набори бібліотечних функцій та типів даних;

- швидкий час компіляції та виконання;

- автоматичне виділення пам'яті та збір сміття;

Недоліки:

- забезпечує меншу гнучкість, ніж C++.

В якості мови програмування для проектування автоматизованої системи обрано C++, тому що вона є сучасною, гнучкою, потужною мовою програмування.

Також було застосовано Windows Forms. Підхід до розробки прикладних програм на Windows Forms ґрунтується на графічному інтерфейсі GDI (Graphics Device Interface, Graphical Device Interface) - це інтерфейс Windows для представлення графічних об'єктів та передачі їх на пристрої відображення, такі як монітори та принтери.

GDI відповідає за відображення ліній та кривих, відображення шрифтів та обробку палітри. Не відповідає за відображення вікон, меню і т.п., це завдання закріплене за підсистемою користувача, що розташовується в user32.dll і що базується на GDI.

Одна з переваг Windows Forms - в тому, що на ньому можна писати косплатформні програми. Проекти, написані на Windows Forms, можна досить легко перенести в іншу операційну систему, якщо на ній встановлений .Net Framework потрібної моделі, де написаний проект.

Також застосовано нейронну мережу для інтелектуалізації автоматизованої системи. Нейронна мережа - це метод прогнозної аналітики,. Система намагається передбачити значення так, як це зробив би людський мозок.

Нейронна мережа працює шляхом створення мережі вхідних вузлів (яка є початком мережі), вихідних вузлів (які показують результати / прогнози, коли дані пройшли через мережу) і прихований шар між цими вузлами.

Прихований шар між вузлами введення і виведення - ось що робить нейронну мережу такою унікальною та ефективною. Кожен раз, коли в нейронну мережу «завантажуються» дані, алгоритм включає дані, які

проходять через неї, привласнюючи «ваги» вузлів в прихованому шарі, що може змінити результат в вихідних вузлах.

Висока точність нейронних мереж піднімає питання, чому вони не використовуються частіше, ніж зараз. Як і очікувалося, у нейронних мереж є й недоліки. Нейронні мережі вимагають більшої обчислювальної потужності, ніж звичайні інструменти прогнозування, що робить їх більш дорогими. Проте для точності прогнозування вразливості застосування нейронних мереж є актуальним.

Призначення автоматизованої системи тестування безпеки веб-додатків - сканування сайту на вразливості, визначення ймовірності вразливості та визначення вразливих місць.

Розроблена автоматизована система тестування структурно складається з 2-х button, 5 label, 2-х textbox та 1 web-Browser

Призначення першої кнопки - для запуску аналізу тексту, призначення другої кнопки - для завантаження веб-сторінки.

Призначення першого textbox - запис адресного рядка. Другий textbox призначений для зберігання звітності

Весь процес сканування веб-додатку відбувається у MyForm.h. До нього підключений заголовний файл і клас myNeuro.h і myNeuro.cpp, де зберігається реалізація алгоритму нейромережі для аналізу тексту.

Аналіз тексту відбувається за допомогою методів нейронної мережі. Алгоритм – без вчителя. У навчанні без вчителя (unsupervised learning) модель має набір даних, і немає явних вказівок, що з даним набором робити. Нейронна мережа намагається самостійно знайти кореляції у даних, витягуючи корисні ознаки та аналізуючи їх.

Текст сторінки записується в рядок, а потім через виклик методу класу відбувається аналіз даного рядка, після аналізу вказується можливий відсоток шкідливого контенту.

3.2 Дослідження розробленої системи і аналіз отриманих результатів

Виконаємо тестування розробленої автоматизованої системи. В якості об’єкту тестування оберемо сайт wikipedia.org.

Результат сканування сайту наведений на рисунку 3.1.

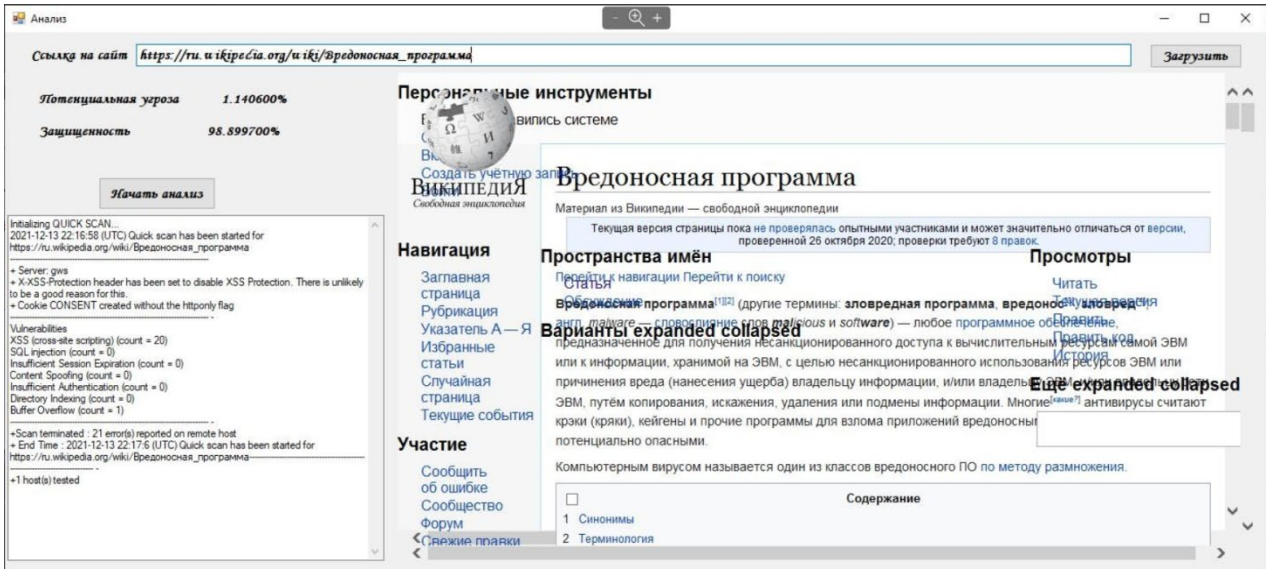

Рисунок 3.1 – Результат сканування сайту

Як бачимо з наведених результатів сканування, захищеність веб-додатку складає 98,9%, потенційна загроза складає 1,1%. Також визначено вразливі місця. Таким чином, система автоматизованого тестування безпеки веб-додатків працює справно, що показує можливість її практичного застосування.

3.3 Висновки до третього розділу

Призначення розробленої автоматизованої системи тестування безпеки веб-додатків - сканування сайту на вразливості, визначення ймовірності вразливості та визначення вразливих місць. Тестування розробленої системи показало її працездатність, дана система, на відміну від своїх аналогів, визначає відсоток захищеності за застосуванням нейронних мереж, що підвищує ефективність тестування та інформаційну безпеку веб-додатку.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Вимоги безпеки під час виконання робіт за ПК

В межах даного розділу роботи розглянемо вимоги безпеки під час виконання робіт за ПК, так як система система автоматизованого тестування безпеки веб-додатків розроблюється безпосередньо з застосуванням ПК.

Робота за комп'ютером передбачає ряд шкідливих факторів та загроз. Що пов'язано з можливістю отримання травм та професійних захворювань, то закон України «Про охорону праці» від 21.11.02 р визначає основні положення відносно реалізації конституційного права громадян на охорону положення відносно реалізації конституційного права громадян на охорону їх життя та здоров'я в процесі трудової діяльності, регулює за допомогою відповідних органів державної влади відносини між роботодавцем і працівником з питань безпеки, гігієни працівника виробничого середовища та встановлює єдиний порядок організації охорони праці в Україні.

Дана робота розроблялась на робочому місці, яке знаходиться на третьому поверсі триповерхової адміністративної будівлі.

Площа приміщення складає 50 $м^2$, що відповідає санітарним нормам, згідно з якими норма на одного працюючого повинна бути не менш 6 $м^2$.

Висота приміщення 3,0 м. таким чином обсяг приміщення складає 150 $м^3$, по нормам – не менш 20 $м^3$ Перелік шкідливих та небезпечних факторів, які діють при роботі на ПЕОМ наведений в табл. 4.1 згідно з та ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [14] та ДСН 33.6.042-99 [15].

Таблиця 4.1 – Шкідливі та небезпечні фактори виробничого середовища

Найменування факторів	Джерела їх виникнення	Параметр, що нормується, та нормативне значення
1	2	3
1. Рівень електромагнітного випромінювання	ЕПТ монітора, системний блок, мережа живлення	Відстань - 50 см, навколо ПК 2-5 кГц – 25 В/м [14]
2. Ультрафіолетове випромінювання	Комп'ютер	Щільність потоку ультрафіолетового випромінювання 10 Вт/м [14]
3. Емоційні перенавантаження, напруга зорового аналізатору	Складність виконання завдань	Зниження реакції користування на звук і світло на 40-50 % [14]
4. Підвищений рівень шуму	Вентилятор, система освітлення, друкувальні прилади	Рівень звуку LA=50 дБ (А) [14]
5. Підвищене значення напруги в електричній мережі	Блок живлення	I=0,6 мА; U=36 В [17]
6. Недолік природнього освітлення	Неправильне планування розташування комп'ютера	КПО, №, Е, лк [14]
7. Вібрація	Вентиляційна система	Віброприскорення, м/с^2, віброшвидкість, м/с або їх рівні LA, LV, дБ [14]
8. Виробничий пил	Статична електрика, накопичена на поверхні комп'ютера	ГДК=4 мг/м^3 [16]
9. Несприятливі температури мікроклімату	Не задовільна робота опалення або вентиляції	Температура (t, °С), вологість (φ, %), швидкість руху повітря (V, м/с). [15]

Оптимальні параметри мікроклімату встановлюються залежно від категорії робіт по фізичному навантаженню [14]. Забезпечення необхідних параметрів мікроклімату досягається у холодний період кондиціюванням та системою опалення, а в теплий період лише системою кондиціювання, згідно з ДБН В.2.5.- 67: 2013 [18].

Таблиця 4.2 – Оптимальні параметри мікроклімату

Категорія робіт по вазі	Період року	Температура, °С	Відносна вологість, %	Швидкість руху повітря, м/с
Легка – Іа	теплий	23-25	40-60	0,1-0,2
	холодний	22-24	40-60	0,1

Освітлення виробничих, службових і допоміжних приміщень регламентується ДБН В.2.5.-28-2006.

Штучне та природне освітлення.

В даному випадку використовується комбіноване освітлення: природне бокове вранці та штучне ввечері.

Виконувана робота відноситься до III розряду зорової праці. Мінімальній розмір об'єктів від 0,3 до 0,5 мм, фон світлий, контраст великий, підрозряд зорових робіт – «Г» в приміщенні забезпеченому комбінованим освітленням: у світлий час доби – бокове однобічне природне освітлення – три віконних прорізи, у темний час загальне чи місцеве рівномірне штучне.

В табл. 4.3 наведені норми освітлення для даного розряду і точності зорових робіт.

Таблиця 4.3 – Характеристики виробничого освітлення

Точність зорових робіт	Мінімальний розмір об'єкта розрізнення, мм	Розряд зорових робіт	Характеристика типу фону	Контраст об'єкта розрізнення з фоном	Розряд зорових робіт	Нормативне значення параметрів освітлення	
						Природне, %	Штучне, лк
Високої точності	0,3-0,5	III	Світлий	Великий	Г	1,2	400

Джерелом шуму в кабінеті може слугувати така техніка: телефон, принтер, факс, комп'ютер, кондиціонер. Рівень шуму не повинен

перевищувати 50 дБ. Рівень вібрації в умовах комфортної роботи, не повинен перевищувати 75 дБ.

Для зменшення рівня звуку та вібрації застосовуються демпфуючі матеріали.

Основними методами захисту є: зниження шуму та вібрації в джерелі (підставки, шумопоглинальні корпуси) і на шляху розповсюдження (ширми, шумопоглинальні стійки), застосування індивідуальних засобів та організаційно-профілактичних методів захисту.

Для захисту від електромагнітного випромінювання застосовується спеціальне покриття екрану. Напруга електромагнітних полів у діапазоні 1 – 12 кГц, 60 – 300кГц по магнітній і електричній складовій повинні відповідати вимогам до ДСанПіН 3.3.2-007-98 [14].

Повітря зовнішнього середовища містить (табл. 4.4) [14].

Таблиця 4.4 – Рівень іонізації повітря при роботі на ПЕОМ

Рівні	Кількість іонів в 1 $см^3$ повітря	
	Позитивні	Негативні
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально припустимі	50000	50000

Напруга електромагнітних полів у діапазоні 1 – 12 кГц, 60 – 300 кГц по магнітній і електричній складовій повинні відповідати вимогам до ДСанПіН 3.3.2-007-98 [14].

При проектуванні систем електропостачання, монтаж силового електроустаткування й електричного освітлення в будинках і приміщеннях для ЕОМ необхідно дотримуватися вимог нормативно-технічної документації.

Комплекс необхідних заходів щодо техніки безпеки визначається, виходячи з видів електроустановки, її номінальної напруги, умов середовища, типу приміщення й доступності електроустаткування.

Документом ПУЕ 2017 [17] передбачені наступні міри електробезпеки:

1) конструктивні заходи: персональна ЕОМ відноситься як електроустановка до 1000 В закритого виконання, усі рубильники встановлені в закритих корпусах, усі струмоведучі частини розміщені в захисних коробах або покриті шаром ізоляції, який виключає можливість дотику до них. Комп'ютер має робочу ізоляцію і елементи заземлення;

2) експлуатаційні міри: при роботі на ЕОМ необхідно дотримувати правила техніки безпеки при роботі з високою напругою, не підключати і не відключати кабелі при включеній напрузі мережі, технічне обслуговування і ремонт проводити тільки при вимкнутому живленні [17];

3) схемно – конструктивні міри: в електричних мережах із глухо заземленим нейтральним провідником як схемно – конструктивну міру безпеки застосовують занулення – навмисне з'єднання металевих неструмоведучих частин комп'ютера з нейтральним провідником [17].

Відповідно до вимог ДБН В 1.1.7-2016 [19] пожежна безпека забезпечується наступними заходами, які застосовуються до категорії В: системою пожежного захисту; організаційними заходами щодо пожежної безпеки; системою запобігання пожеж: запобігання утворенню горючого середовища, та запобігання утворення у горючому середовищі джерел запалювання.

Для зменшення небезпеки утворення в стальному середовищі джерел запалювання передбачено:

1) використання устаткування, що відповідає класу пожежобезпечної зони ПІІа: ступінь захисту електроапаратури повинна бути не менш IP-44, ступінь захисту світильників IP-23, відповідно до ДБН В 1.1.7-2016 [19];

2) блискавковідвід будинків, споруджень і устаткування для даного класу пожежонебезпеки, зони П-ІІа і місцевості із середньою грозовою

діяльністю 20 і більше грозових годин у рік, тобто встановлена III категорія блискавко захисту відповідно до ДСТУ EN 62305-1:2012 [20];

3) застосування заземлення захисного екрану для стоку статичної електроенергії; використання для гасіння пожежі вуглекислого вогнегасника ВВ-2.

Організаційними заходами протипожежної профілактики є: навчання виробничого персоналу протипожежним правилам; видання необхідних інструкцій, плакатів, засобів наочної агітації, плану евакуації персоналу у випадку пожежі.

4.2 Дії персоналу в надзвичайних ситуаціях

Розглянемо дії персоналу у різних надзвичайних ситуаціях. Дії у разі вибуху.

Вибух. Основні вражаючі фактори вибуху: повітряна ударна хвиля та уламкові поля, що утворюються уламками зруйнованих об'єктів, що летять, технологічного обладнання, вибухових пристроїв

При загрозі вибуху слід лягти на живіт, захищаючи голову руками, подалі від вікон, засклених дверей, проходів, сходів.

Якщо стався вибух, вжити заходів щодо недопущення пожежі та паніки; надати першу допомогу постраждалим. Кожен працівник при виявленні вогнища або ознак горіння (задимлення, запах гару, підвищення температури тощо) повинен негайно повідомити про це за телефоном «101».

При цьому назвати найменування об'єкта, місце вибуху, пожежі, а також своє прізвище; вжити заходів щодо евакуації людей, гасіння пожежі та збереження матеріальних цінностей.

Вимоги щодо використання первинних засобів пожежогасіння: Вуглекислотні вогнегасники (ОУ-2, ОУ-3, ОУ-5, ОУ-6, ОУ-7 тощо) призначені для гасіння загорянь різних горючих речовин, виключення, коли горіння яких відбувається без доступу повітря,

Для приведення в дію вуглекислотних вогнегасників необхідно розтруб направити на палаючий предмет, зірвати пломбу, висмикнути чеку, натиснути на важіль (або повернути маховик вентиля вліво вщент), направити струмінь на полум'я. Тримати вогнегасник вертикально, перевертати його не потрібно.

Щоб уникнути обморожування, не торкатися металевої частини розтрубу. оголеними частинами тіла. При гасінні електроустановок, що знаходяться під напругою, не допускається підводити до них розтруб ближче 1м.

Внутрішні пожежні крани призначені для подачі води.

Асбестове полотно, повсть (кошма) використовуються для гасіння невеликих вогнищ загоряння будь-яких речовин та матеріалів, горіння яких не може відбуватись без доступу повітря.

Дії у разі хімічної аварії.

Небезпека хімічної аварії для людей полягає в порушенні нормальної життєдіяльності організму та можливості віддалених генетичних наслідків, а за певних обставин – у летальному результаті при потраплянні АХНР в організм через органи дихання, шкіру, слизові оболонки, рани та разом з їжею.

При отриманні сигналу про хімічну аварію включити радіоприймач для отримання достовірної інформації про аварію та рекомендовані дії.

Закрити вікна, вимкнути електропобутові прилади. Для захисту органів дихання використовувати ватно-марлеву пов'язку або підручні вироби з тканини, змочені у воді, 2-5%-ному розчині харчової соди (для захисту від хлору), 2%-ному розчині лимонної або оцтової кислоти (для захисту від аміаку).

При неможливості залишити зону зараження щільно закрити двері, вікна, вентиляційні отвори та димарі; щілини в них заклеїти папером чи скотчем.

Не ховатися на перших поверхах будівель, у підвалах та напівпідвалах.

При підозрі на поразку АХНР виключити будь-які фізичні навантаження, прийняти рясне питво (молоко, чай) та негайно звернутися до лікаря.

Утримуватися від вживання водопровідної води – до офіційного висновку щодо її безпеки. На зараженій місцевості рухатися швидко, але не бігти, піднімаючи пил, не торкатися навколишніх предметів, не наступати пролиту рідину або порошкоподібні розсипи невідомих речовин.

Дії у разі обвалення будівель, споруд. Повне або часткове раптове обвалення будівлі – це надзвичайна ситуація природного або техногенного характеру, яка також виникає за причини помилок, допущених на етапі проектування.

Руйнування комунально-енергетичних мереж, утворення завалів, травмуванню та загибелі людей. Почувши вибух або виявивши, що будівля втрачає свою стійкість, негайно покинути його.

Залишаючи приміщення, спускатися сходами, а не на ліфті: він у будь-який момент може зупинитись.

Не панікувати, не влаштовувати тисняву у дверях під час евакуації. Зупиняти тих, хто збирається стрибати з балконів (поверхів вище першого) та через засклені вікна. Якщо відсутня можливість покинути будівлю, зайняти безпечне місце: отвори капітальних внутрішніх стін, кути, утворені капітальними внутрішніми стінами, під балконами каркасу (вони захищають від падаючих предметів та уламків). Відкрити двері з приміщення, щоб забезпечити вихід.

Не піддаватися паніці та зберігати спокій. Триматися подалі від вікон, електроприладів.

Якщо виникла пожежа, негайно спробувати загасити її. Телефон використовувати лише для виклику представників правоохоронних органів, пожежної охорони, лікарів, рятувальників

Не користуватися сірниками: існує небезпека вибуху внаслідок витоку газу. Опинившись надворі, не стояти поблизу будинку. Перейти на відкритий простір.

Дії у разі знаходження під завалом:

Дихати глибоко, не піддаватися паніці.

По можливості надати собі першу допомогу. Пристосуватися до обстановки та озирнутися, пошукати вихід. Спробувати визначити, де знаходиться людина, чи немає інших людей: прислухатись, подати голос.

Слід пам'ятати: людина здатна витримати спрагу і голод протягом тривалого часу, якщо не марно витрачати енергію.

Пошукати в кишенях або поблизу предмети, щоб подати світлові або звукові сигнали: ліхтарик або металеві предмети, якими можна постукати по трубі чи стіні (привернути увагу рятувальників).

Якщо єдиним виходом є вузький лаз - протиснутися через нього. Для цього розслабити м'язи та рухатися, притиснувши лікті до тіла.

Також розглянемо дії при електротравматизмі. Згідно з наказом № 398 «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою» [21], заходи першої допомоги залежать від стану потерпілого після визволення його від електричного струму. Для визначення стану необхідно вжити таких заходів:

- покласти потерпілого спиною на тверду поверхню;

- перевірити наявність у потерпілого дихання;

- перевірити наявність у потерпілого пульсу на сонній артерії;

- з'ясувати стан зіниці, широка зіниця вказує на погіршення кровопостачання.

У всіх випадках ураження електричним струмом виклик лікаря є обов'язковим незалежно від стану потерпілого.

Якщо потерпілий знаходиться при свідомості, його треба покласти у зручне положення і до прибуття лікаря забезпечити спокій, обов'язково спостерігаючи задиханням і пульсом.

Не можна дозволяти потерпілому рухатись, продовжувати роботу. Якщо лікаря швидко викликати не можна, необхідно терміново доставити потерпілого у медичний пункт.

Якщо потерпілий знаходиться у непритомному стані, його необхідно покласти, розстебнути одяг, забезпечити приплив свіжого повітря, дати понюхати нашатирний спирт, бризнути на нього водою і забезпечити спокій. У той же час потрібно викликати лікаря. Якщо потерпілий дихає погано, рідко і судомно, йому необхідно робити штучне дихання і непрямий масаж серця.

У разі відсутності в потерпілого ознак життя не можна вважати його померлим. Якщо в такому стані потерпілому не буде надано негайну першу допомогу у вигляді штучного дихання і зовнішнього масажу серця, то настане смерть.

Оживлення організму, ураженого електричним струмом, може бути проведено кількома способами. Всі вони базуються на штучному диханні. Починати штучне дихання слід негайно після вивільнення потерпілого від електричного струму і проводити безперервно до досягнення позитивного результату.

Штучне дихання необхідно робити безперервно, до прибуття лікаря.

Переносити потерпілого до іншого місця треба тільки в тих випадках, коли йому, чи особі, яка надає допомогу, продовжує загрожувати небезпека.

Ураженого електричним струмом можна визнати померлим тільки за наявності видимих тяжких зовнішніх ушкоджень: роздроблення черепа у разі падіння чи обпалення всього тіла.

В інших випадках констатує смерть лише в лікарні.

Також розглянемо дії при пожежі. Про виникнення пожежі в приміщеннях негайно повідомити пожежну охорону.

При цьому необхідно назвати адресу, зазначити кількість поверхів будівлі, місце виникнення пожежі, обстановку на пожежі, наявність людей, а також повідомити своє прізвище.

Вжити (по можливості) заходи на евакуацію людей, гасіння (локалізацію) пожежі з використанням первинних засобів пожежогасіння та на збереження матеріальних цінностей.

Повідомити про виникнення пожежі керівника (заступників керівника) чи відповідальну компетентну посадову особу та чергового охорони.

У разі необхідності, викликати інші аварійно-рятувальні служби (медичну, газорятувальну тощо).

ВИСНОВКИ

В результаті написання магістерської роботи проведено дослідження та проектування автоматизованої системи тестування безпеки веб-додатків. Встановлено, що тестування використовується веб-розробниками та адміністраторами безпеки для тестування та вимірювання рівня безпеки веб-додатка з використанням ручних та автоматизованих методів тестування безпеки. Основна мета тестування безпеки веб-програм - виявити будь-які вразливості або загрози, які можуть поставити під загрозу безпеку або цілісність веб-програми.

В ході написання роботи встановлено, що існуючі системи автоматизованого тестування не є в достатній мірі ефективними та функціональними щодо виявлення сучасних загроз та вразливостей, саме тому в роботі проведена розробка власної системи тестування. Аналіз тексту в системі відбувається за допомогою методів нейронної мережі. Текст сторінки записується в рядок, а потім через виклик методу класу відбувається аналіз даного рядка, після аналізу вказується можливий відсоток шкідливого контенту. Даний підхід є ефективним та функціональним, що дозволяє підвищити інформаційну безпеку веб-додатків.

В результаті написання роботи були виконані наступні задачі:

1. Виконано огляд літератури з теми дослідження. Досліджено проблеми захисту веб-додатків та визначено актуальність автоматизованого тестування;

2. Здійснено огляд засобів захисту веб-додатків та розглянуто системи автоматизованого тестування;

3. Виконано розробку та дослідження системи автоматизованого тестування безпеки веб-додатків.

4. Розглянуто питання з охорони праці та безпеки у надзвичайних ситуаціях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Безопасность веб-приложений: от уязвимостей до мониторинга [Електронний ресурс]: Режим доступу: https://habr.com/en/ company/pentestit/blog/526878/ (дата звернення: 01.11.2021).

2. О.Бондаренко, І.Ушкалено. Безпека web-додатків: Актуальні проблеми та їх аналіз. Формування ринкової економіки в Україні. 2017. Вип. 38. С. 28-36.

3. Бабаш, А.В. Информационная безопасность: Лабораторный практикум / А.В. Бабаш, Е.К. Баранова, Ю.Н. Мельников. - М.: КноРус, 2019. - 432 с.

4. The Web Application Security Consortium [Електронний ресурс]: Режим доступу: https://www. academia. Edu /11623665/ Web_ Application_ Security_ Consortium_ Threat_ Classification_WASC-TC_and_ISECOM_ Open_ Source_ Security_ Testing_ Methodology_Manual_OSSTMM_ (дата звернення: 07.11.2021).

5. Гришина, Н.В. Информационная безопасность предприятия: Учебное пособие / Н.В. Гришина. - М.: Форум, 2018. - 118 с.

6. 2021 CWE Top 25 Most Dangerous Software Weaknesses [Електронний ресурс]: Режим доступу: http://cwe.mitre.org/ top25/archive/2021/ 2021_cwe_top25.html (дата звернення: 11.11.2021).

7. OWASP Top 10:2021 [Електронний ресурс]: Режим доступу: https://owasp.org/Top10/ (дата звернення: 15.11.2021).

8. OWASP Top 10:2021 List [Електронний ресурс]: Режим доступу: https://owasp.org/Top10/0x00-notice/ (дата звернення: 20.11.2021).

9. OWASP Top 10: 10 самых опасных уязвимостей [Електронний ресурс]: Режим доступу: https://cisoclub.ru/owasp-top-10-10-samyh-opasnyh-uyazvimostej/ (дата звернення: 22.11.2021).

10. Партыка, Т.Л. Информационная безопасность: Учебное пособие / Т.Л. Партыка, И.И. Попов. - М.: Форум, 2018. - 88 с.

11. Бенкен, Елена AJAX. Программирование для Интернета (+ CD-ROM) / Елена Бенкен , Геннадий Самков. - М.: БХВ-Петербург, 2017. - 464 с.

12. Будилов, В. Основы программирования для Интернета. Самоучитель / В. Будилов. - М.: БХВ-Петербург, 2016. - 634 с.

13. Баранова, Е.К. Информационная безопасность и защита информации: Учебное пособие / Е.К. Баранова, А.В. Бабаш. - М.: Риор, 2017. - 400 с.

14. ДСанПІН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. Затверджено Постановою Головного державного санітарного лікаря України 10 грудня 1998 р. №7.

15. ДСН 3.3.6.037-99. Державні санітарні норми. Шум. Загальні вимоги безпеки. – К.: Затверджено Постановою Головного державного санітарного лікаря України 1 грудня 1999 р. №37.

16. ДБН В.2.5.-28-2006. Державні будівельні норми. Природне і штучне освітлення. – К.: Інститут "Київпромелектропроект". Наказ від 30.12.2011 № 438.

17. Правила улаштування електроустановок. – Видання офіційне. Міненерговугілля України. – Х.: Видавництво «Форт», 2017. – 760 с.

18. ДБН В.2.5-67:2013 Опалення, вентиляція та кондиціонування. Наказ 25.01.2013 № 24.

19. ДБН В.1.1.7-2016. Захист від пожежі. Пожежна безпека об'єктів будівництва. Наказ від 31.10.2016 № 287.

20. ДСТУ EN 62305-1:2012 Захист від блискавки. Частина 1. Загальні принципи (EN 62305-1:2011, IDT). Наказ від 28.05.2012 № 640 Про прийняття міжнародних та європейських нормативних документів як національних нормативних документів методом підтвердження».

21. Наказ від 16.06.2014 № 398 «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою».

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ

```
#pragma once
#include <ctime>
#include <math.h>
#include "stdlib.h"
#include "myNeuro.h"
#include <msclr\marshal_cppstd.h>
using namespace std;
namespace site {
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
/// <summary>
/// </summary>
public ref class MyForm : public
System::Windows::Forms::Form
{
public:
MyForm(void)
{
InitializeComponent();
}
protected:
/// <summary>
/// </summary>
```

```
    ~MyForm()
    {
    if (components)
    {
    delete components;
    }
    }
    private: System::Windows::Forms::Button^ button1;
    protected:
    private: System::Windows::Forms::Label^ label1;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::Label^ label4;
    private: System::Windows::Forms::Label^ label5;
    private: System::Windows::Forms::RichTextBox^
richTextBox1;
    private:
    /// <summary>
    /// </summary>
    System::ComponentModel::Container ^components;
    #pragma region Windows Form Designer generated code
    /// <summary>
    /// </summary>
    void InitializeComponent(void)
    {
    System::ComponentModel::ComponentResourceManager^
resources = (gcnew
System::ComponentModel::ComponentResourceManager(MyForm
::typeid));
```

```
    this->button1 = (gcnew
System::Windows::Forms::Button());
    this->label1 = (gcnew
System::Windows::Forms::Label());
    this->label2 = (gcnew
System::Windows::Forms::Label());
    this->label3 = (gcnew
System::Windows::Forms::Label());
    this->label4 = (gcnew
System::Windows::Forms::Label());
    this->label5 = (gcnew
System::Windows::Forms::Label());
    this->richTextBox1 = (gcnew
System::Windows::Forms::RichTextBox());
    this->SuspendLayout();
    //
    // button1
    //
    this->button1->Font = (gcnew
System::Drawing::Font(L"Monotype Corsiva", 12,
static_cast<System::Drawing::FontStyle>((System::Drawin
g::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
    System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
    this->button1->Location =
System::Drawing::Point(82, 127);
    this->button1->Name = L"button1";
    this->button1->Size = System::Drawing::Size(131,
36);
```

```
    this->button1->TabIndex = 0;
    this->button1->Text = L"Начать анализ";
    this->button1->UseVisualStyleBackColor = true;
    this->button1->Click += gcnew
System::EventHandler(this, &MyForm::button1_Click);
    //
    // label1
    //
    this->label1->AutoSize = true;
    this->label1->Font = (gcnew
System::Drawing::Font(L"Monotype Corsiva", 12,
static_cast<System::Drawing::FontStyle>((System::Drawin
g::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
    System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
    this->label1->Location = System::Drawing::Point(12,
30);
    this->label1->Name = L"label1";
    this->label1->Size = System::Drawing::Size(164,
18);
    this->label1->TabIndex = 1;
    this->label1->Text = L"Потенциальная угроза";
    //
    // label2
    //
    this->label2->AutoSize = true;
    this->label2->Font = (gcnew
    System::Drawing::Font(L"Monotype Corsiva", 12,
static_cast<System::Drawing::FontStyle>((System::Drawin
```

```
g::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
    System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
    this->label2->Location =
System::Drawing::Point(540, 9);
    this->label2->Name = L"label2";
    this->label2->Size = System::Drawing::Size(160,
18);
    this->label2->TabIndex = 2;
    this->label2->Text = L"Введите текст сайта";
    //
    // label3
    //
    this->label3->AutoSize = true;
    this->label3->Font = (gcnew
System::Drawing::Font(L"Monotype Corsiva", 12,
static_cast<System::Drawing::FontStyle>((System::Drawin
g::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
    System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
    this->label3->Location =
System::Drawing::Point(216, 30);
    this->label3->Name = L"label3";
    this->label3->Size = System::Drawing::Size(28, 18);
    this->label3->TabIndex = 3;
    this->label3->Text = L"0%";
    //
    // label4
```

```
    //
    this->label4->AutoSize = true;
    this->label4->Font = (gcnew
System::Drawing::Font(L"Monotype Corsiva", 12,
static_cast<System::Drawing::FontStyle>((System::Drawin
g::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
    System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
    this->label4->Location =
System::Drawing::Point(200, 66);
    this->label4->Name = L"label4";
    this->label4->Size = System::Drawing::Size(44, 18);
    this->label4->TabIndex = 4;
    this->label4->Text = L"100%";
    //
    // label5
    //
    this->label5->AutoSize = true;
    this->label5->Font = (gcnew
System::Drawing::Font(L"Monotype Corsiva", 12,
static_cast<System::Drawing::FontStyle>((System::Drawin
g::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
    System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
    this->label5->Location = System::Drawing::Point(12,
66);
    this->label5->Name = L"label5";
```

```
    this->label5->Size = System::Drawing::Size(109,
18);
    this->label5->TabIndex = 5;
    this->label5->Text = L"Защищенность";
    //
    // richTextBox1
    //
    this->richTextBox1->Location =
System::Drawing::Point(304, 30);
    this->richTextBox1->Name = L"richTextBox1";
    this->richTextBox1->Size =
System::Drawing::Size(603, 350);
    this->richTextBox1->TabIndex = 7;
    this->richTextBox1->Text = resources-
>GetString(L"richTextBox1.Text");
    //
    // MyForm
    //
    this->AutoScaleDimensions =
System::Drawing::SizeF(6, 13);
    this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
    this->ClientSize = System::Drawing::Size(933, 410);
    this->Controls->Add(this->richTextBox1);
    this->Controls->Add(this->label5);
    this->Controls->Add(this->label4);
    this->Controls->Add(this->label3);
    this->Controls->Add(this->label2);
    this->Controls->Add(this->label1);
    this->Controls->Add(this->button1);
```

```
    this->Name = L"MyForm";
    this->Text = L"Анализ";
    this->ResumeLayout(false);
    this->PerformLayout();
            };
    #pragma endregion
    private: System::Void button1_Click(System::Object^
sender, System::EventArgs^ e) {
    string k1 =
msclr::interop::marshal_as<std::string>(richTextBox1-
>Text);
    myNeuro* bb = new myNeuro();
    //----------------------------------INPUTS----
GENERATOR-------------
    srand(time(NULL));
    float* abc = new float[100];
    for (int i = 0; i < 100; i++)
    {
    abc[i] = (rand() % 98) * 0.01 + 0.01;
    }
    float* cba = new float[100];
    for (int i = 0; i < 100; i++)
    {
    cba[i] = (rand() % 98) * 0.01 + 0.01;
    }
    //----------------------------------TARGETS----
GENERATOR-------------
    float* tar1 = new float[2];
    tar1[0] = 0.01;
    tar1[1] = 0.99;
```

```
    float* tar2 = new float[2];
    tar2[0] = 0.99;
    tar2[1] = 0.01;
    //--------------------------------NN---------
WORKING---------------
    bb->query(abc);
    bb->query(cba);
    int i = 0;
    while (i < 100000)
    {
    bb->train(abc, tar1);
    bb->train(cba, tar2);
    i++;
    }
    //cout << "_____________________RESULT_____________";
    bb->query(abc);
    label3->Text = gcnew System::String(bb-
>GetStr11().c_str());
    bb->query(cba);
    label4->Text = gcnew System::String(bb-
>GetStr12().c_str());
    //cout <<
"_________________THE____END_________________";
    int l = rand()%20+15;
    int count = 0;
    vector<int> arr;
    int a1, a2;
    int counter = l;
    int f = 0;
    for (int i = 0; i < k1.size(); ++i)
```

```
{
if (k1[i] == ' ' || k1[i]=='\n')
count++;
if (count == counter && f == 0)
{
a1 = i;
f = 1;
}
if (count == counter + 1)
{
a2 = i;
counter += l;
arr.push_back(a1);
arr.push_back(a2-a1);
f = 0;
}
}
int c = 0;
for (int i = 0; i < arr.size(); i += 2) {
richTextBox1->Select(arr[i], arr[i+1]);
richTextBox1->SelectionColor = Color::Red;
richTextBox1->Refresh();
}
}
};
}
```