МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Експертна система тестування безпеки веб-ресурсів»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Лаврик Т.В.**

Студента групи КБ – 61 — **Забара В.М.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2012 г.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи КБ-61 спеціальності "Кібербезпека" денної форми навчання Забари Валентина Михайловича.

Тема: " Експертна система тестування безпеки веб-ресурсів "

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: 1) аналітичний огляд загроз та методик тестування безпеки веб-ресурсів; 2) постановка завдання й формування завдань дослідження; 3) характеристика експертних систем для розв'язування задач інформаційної безпеки; 5) розробка інформаційного й програмного забезпечення експертної системи; 6) аналіз результатів експерименту.

Дата видачі завдання "_____"_____2020 р.

Керівник випускної роботи _____ Лаврик Т.В.

Завдання прийняв до виконання _____ Забара В.М.

РЕФЕРАТ

Записка: 58 сторінок, 7 рисунків, 6 таблиць, 1 додаток, 17 джерел.

Об'єкт дослідження – слабоформалізований процес тестування безпеки веб-ресурсів.

Мета роботи – розробка експертної системи для тестування безпеки веб-ресурсів.

Методи дослідження – метод аналітичного огляду, теорія систем штучного інтелекту, моделі та методи представлення знань в експертних системах.

Результати – сформовано вхідний математичний опис експертної системи; створена база знань, яка містить опис основних етапів тестування веб-ресурсів і способів їх виявлення у вигляді продукційних правил; розроблено і програмно реалізовано алгоритми зберігання і обробки таких правил; працездатність експертної системи перевірена на прикладі діагностики веб-ресурсу.

ВЕБ-РЕСУРС, ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-РЕСУРСА,
ЕКСПЕРТНА СИСТЕМА, ПРОДУКЦІЙНІ ПРАВИЛА,
БАЗА ЗНАНЬ.

ЗМІСТ

ВСТУП

Вразливості, як і раніше залишаються одним із найбільш поширених недоліків в сфері забезпечення захисту інформації в веб-ресурсі. Інші проблеми, які часто зустрічаються – це низький рівень знань працівників щодо інформаційної безпеки, надання невеликого значення парольній політиці, відсутність контролю за виконанням парольної політики, збій через дефекти в процесі оновленнями програмного забезпечення, конфігурації, які не відповідають стандартам безпеки, розмежування прав доступ, які не являються ефективними.

Питання вразливості веб-ресурсів багато разів було розглянуте в спеціалізованих роботах та науково-популярних статтях. Але серед цих матеріалів на даний момент важко знайти конкретні засоби та механізми попереднього виявлення вразливостей, які допоможуть знизити їх кількість до початку експлуатації веб-ресурсів. Ця проблема стає складнішою за рахунок того, що багато розробників нехтують захистом заради чогось іншого або забувають надати цьому велике значення. При цьому проблеми безпеки інформації, які створюються, показують свій негативний вплив вже на етапі експлуатації замовником. Це породжує додаткові витрати, які могли бути мінімальними, якщо дана вразливість була виявлена на етапі розробки.

Недооцінка важливості безпеки веб-ресурсів і є головною причиною низького рівня захисту веб-ресурсів.

1 ОСОБЛИВОСТІ ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-РЕСУРСІВ

1.1 Аналіз вразливостей та загроз безпеки веб-ресурсів

Відповідно до стандарту ISO/IEC 29147:2018 у контексті інформаційних технологій та кібербезпеці вразливість – це стан або сукупність певних умов, присутніх у системі, продукті, компоненті чи послузі, що порушує неявну або явну політику безпеки [1]. Вразливість може розглядатися як слабке місце, що дозволяє впливати на безпеку. Зловмисники використовують вразливості, щоб порушити конфіденційність, цілісність, доступність, функціонування чи якусь іншу властивість безпеки. Уразливості часто є наслідком збоїв програми чи системи для безпечного оброблення ненадійного або несподіваного вводу. Причини, що призводять до вразливості, включають помилки в кодуванні або конфігурації, недогляди у виборі дизайну, а також незахищені специфікації протоколу та формату. Незважаючи на значні зусилля щодо поліпшення безпеки програмного забезпечення, сучасне програмне забезпечення та системи настільки складні, що виробляти їх без уразливих ситуацій недоцільно. До факторів ризику вразливості належать такі:

— оперуючи та покладаючись на системи, які мають відомі вразливості;

— не мають достатньої інформації про вразливості;

— не знають, що вразливості існують.

Згідно НД ТЗІ 1.1–003–99 вразливість – це нездатність системи протистояти реалізації певних загроз або сукупності таких загроз [2].

Одним із міжнародних класифікацій вразливостей є WASC Threat Classification. WASC (Web Application Security Consortium) – це міжнародне об’єднання з питання безпеки веб–додатків. Це неприбуткова організація, яка складається з експертів, фахівців з інформаційної безпеки та безпеки мережі Інтернет з різних куточків світу [1].

Згідно WASC Threat Classification вразливості веб-додатків поділяються за етапами життєвого циклу програмного забезпечення [3]:

- Вразливості етапу проектування – до цієї категорії вразливостей входять ті

які з’являються через помилки під час проектування веб-ресурса.

- Вразливості етапу реалізації – до цієї категорії вразливостей входять ті які з’являються через помилки під час реалізації окремих елементів веб–ресурса.
- Вразливості етапу розгортання – до цієї категорії вразливостей входять ті які з’являються через полики підчас налаштування веб–ресурса до роботи.

За типом атак вразливості діляться наступним чином:

1. Атака на аутентифікацію або атака на адміністратора. Це атака заснована на отримання прав адміністратора, шляхом аутентифікації під виглядом адміністратора. До цього типу атак можна віднести наступні як брут-форм (повний перебір), недостатня аутентифікацію, небезпечне відновлення паролів.

2. Атака на авторизацію, також відноситься до атаки на адміністратора. Ця атака може бути здійснена шляхом передбачення ідентифікатора сеанса, фіксації сеансу, недостатньої авторизації, або відсутності тайм-аута сеанса.

3. Атак на клієнтів, а саме на їх браузери (підміна змісту, міжсайтовий скриптинг, розщеплення HHTP –запиту).

4. Атаки на сервер, шляхом виконання коду на сервері. До таких атак можно віднести: переповнення буфера, атака на формат рядка SQL ін'єкції, LDAP ін'єкції, виконання команд ОС.

5. Атаки націлені на розробника або логічні атаки на проектувальника. Індексування каталогів, ідентифікація додатків, витік інформації, зворотний шлях в каталогах, передбачуване розміщення ресурсів.

Іншим відомим описом вразливостей є OWASP Top 10. OWASP – це неприбуткова організація, яка працює над підвищенням безпеки програмного забезпечення. OWASP є джерелом для розробників та технологій для захисту Інтернету [4].

OWASP Top 10 – це щорічний мануал для розробників з питань безпеки веб-ресурсів. Він представляє собою список та пояснення щодо найбільш

критичних ризиків та вразливостей для безпеки веб-додатків.

До списку OWASP Top 10 за 2019 рік входять наступні вразливості [4]:

1. Ін’єкції. Можливість до ін'єкції на сайті (SQL, NoSQL, OC, LDAP) зявляються, якщо недостовірні дані які надсилаються інтерпретатору, являються частиною деякої команди чи запиту. Такі дані зловмисників можуть змусити інтерпретатора виконувати ненавмисні команди або допомогти отримати доступ до даних без належного дозволу.

2. Проблеми зі аутентифікацією . Функції програми, пов’язані з аутентифікацією та керуванням сеансом, часто реалізуються неправильно, що дозволяє зловмисникам викрадати паролі, ключі або маркери сеансу або використовувати інші недоліки в реалізації, щоб отримувати доступ до акаунтів адміністратора та інших користувачів тимчасово або назавжди.

3. Незахищеність даних, критичних даних користувачів. Багато веб-додатків та API не захищають належним чином конфіденційні дані. Зловмисники можуть вкрасти або змінити такі слабко захищені дані, щоб вчинити шахрайство з кредитною карткою, крадіжку особи або інший злочин. Чутливі дані можуть бути порушені без додаткового захисту, наприклад шифрування в спокої або під час транзиту, і вимагають спеціальних заходів безпеки при обміні з браузером.

4. Зовнішні об’єкти XML (XXE). На старих веб-ресурсах, та на процесорах-XML з небезпечними конфігураціями, виконують оцінку URL-посилань на сторонні ресурси в XML документах. Ці посилання можуть потенційно можуть бути використаними зловмисниками, для отримання доступу до внутрішніх файлів за допомогою обробника URI, обміну внутрішніми файлами, сканування внутрішніх портів, віддаленого виконання коду та відмови в сервісних атаках.

5. Порушений контроль доступу. Обмеження щодо дозволених користувачів, які мають певні повноваження змінювати в системи, часто не виконуються належним чином. Зловмисники можуть використовувати ці

недоліки для доступу до несанкціонованих функціональних можливостей та / або даних, таких як доступ до облікових записів інших користувачів, перегляд конфіденційних файлів, зміна даних інших користувачів, зміна прав доступу тощо.

6. Неправильна конфігурації безпеки. Ця вразливість безпеки є однією з найчастішою проблемою. Зазвичай це результат небезпечних конфігурацій за замовчуванням, неповних або спеціальних конфігурацій, відкритого хмарного сховища, неправильно налаштованих заголовків HTTP та багатослівних повідомлень про помилки, що містять конфіденційну інформацію. Не тільки всі операційні системи, рамки, бібліотеки та додатки повинні бути надійно налаштовані, але вони повинні бути виправлені / модернізовані своєчасно.

7. Міжсайтовий скриптинг (XSS**).** Вразливість XSS виникають щоразу, коли програма включає недовірені дані на новій веб-сторінці без належної перевірки чи видалення, або оновлює наявну веб-сторінку із наданими користувачем даними за допомогою API браузера, який може створювати HTML або JavaScript. Ця вразливість, дозволяє злочинцю запустити свій скрипт в браузерах жертв. Зазвичай вони перехоплюють сеанси, видаляють контент веб-сайта, або використовуються для переправлення трафіка на інші, часто шкідливі веб-сайти.

8. Небезпепечна десерелізація**.** Вразливість призводить до віддаленого виконання коду. Навіть якщо недоліки десеріалізації не призводять до віддаленого виконання коду, їх можна використовувати для виконання інших атак, включаючи атаки відтворення стороннього контенту, атаки ін'єкції та атаки на зміну привілеїв доступу.

9. Використання компонентів, з відомими вразливими місцями. Компоненти, такі як бібліотеки, рамки та інші програмні модулі, працюють із тими ж привілеями, що і програма. Якщо використовується вразливий компонент, така атака може полегшити серйозну втрату даних або захоплення сервера. Програми та API, що використовують компоненти з відомою

вразливістю, можуть підірвати захисні програми та включити різні атаки та впливи.

10. Недостатній облік даних та моніторинг. Недостатній облік та моніторинг у поєднанні з відсутньою або неефективною інтеграцією з реакцією на інцидент дозволяє зловмисникам надалі атакувати системи, підтримувати стійкість, перемикатись на більшість систем, а також підробляти, витягувати або знищувати дані. Більшість досліджень щодо порушення виявляють час виявлення порушення понад 200 днів, як правило, виявляються зовнішніми сторонами, а не внутрішніми процесами чи моніторингом.

1.2 Огляд існуючих методик тестування безпеки веб-ресурсів

Тестування безпеки – це стратегія для виконання перевірки безпеки системи та аналізу ризиків, які необхідні для забезпечення цілісного підходу до захисту програми від атак хакерів, шкідливого програмного забезпечення, несанкціонованого доступу до конфіденційних даних [5].

За рівнем знань про систему існують наступні підходи до тестування безпеки веб-ресурсів [5, 6]:

- BlackBox («чорний ящик»): виконавець володіє тільки загальнодоступною інформацією про мету дослідження (ім'я компанії або її сайт). Даний варіант максимально наближений до реальної ситуації.
- WhiteBox («білий ящик»): виконавцю надається максимум необхідної для нього інформації (навіть адміністративний доступ на сервера).
- GrayBox («сірий ящик»): виконавець діє за варіантом BlackBox і періодично запитує інформацію про систему, що тестується.

Основним типом тестування безпеки веб-ресурсів є тест на проникнення. Тест на проникнення (англ. penetration test, pentest, тест на подолання захисту) – це метод оцінювання захищеності інформаційної системи (Web-додаток, вебпортал, корпоративна мережа тощо) шляхом моделювання дій зовнішніх зловмисників з проникнення у систему (які не

мають авторизованих засобів доступу до системи) і внутрішніх зловмисників (які мають певний рівень санкціонованого доступу) [5].

Найпоширенішими методологіями проведення тестування на проникнення є такі:

- Open Source Security Testing Methodology Manual (OSSTMM);
- NIST Special Publications 800-115 Technical Guide to Information Security Testing and Assessment;
- Penetration Testing Execution Standard (PTES);
- OWASP Testing Guide;
- Open Source Security Testing Methodology Manual.

OSSTMM – це міжнародна методологія для тестування інформаційної безпеки, яка була створена ISECOM (Institute for Security and Open Methodologies). Методологія має так звану «Карту безпеки». Карта вказує галузі безпеки, які повинні бути протестовані на відповідність методиці [6]:

- Тестування процесу безпеки.
- Тестування технології інтернет-безпеки.
- Тестування безпеки каналів зв'язку.
- Тестування безпеки бездротових технологій.
- Тестування фізичної безпеки.

Дана методологія має детальний опис підготовки для кожного етапу тестування, детальний опис всіх методів та підходи які використовуються тестування, окремо винесені та описані поняття та терміні інформаційної безпеки. Але при цьому методологія не містить опису інструментів, які повинні бути використані для цього.

Penetration Testing Execution Standard (PTES**)** лише на питаннях проведення тестування на проникнення [7].

Дана методологія розділяє тестування на 7 фаз:

1. визначення список інформації необхідної під час тестування;
2. збір всіх доступних даних;

3. визначення напрямок атак;
4. пошук потенційних вразливостей системи;
5. використання знайдених вразливостей для підтвердження можливості обходу засобів захисту;
6. визначення даних які можна отримати про систему, після проникнення;
7. складання звіту.

Дана методологія має добре провірену послідовність дій для тестування на проникнення в мережу, але не приділяється достатньої уваги фізичній безпеці мережі, та питанню використання методів соціальній інженерії.

OWASP Testing Guide детально охоплює питання безпеки Web-сайтів, Web-додатків.OWASP Testing Guide містить практичні рекомендації для проведення тестування на проникнення, а також низькорівневе керівництво, яке описує техніки тестування для найбільш поширених вразливостей у Web-додатках і Web-сервісах [8].

Під час пасивного етапу тестувальник намагається зрозуміти логіку програми і «грає» з ним. Можуть використовуватися інструменти для збору інформації. Наприклад, за допомогою HTTP проксі можна вивчити всі HTTP запити і відповіді. В кінці цього етапу тестувальник повинен розуміти всі крапки входу додатки (наприклад, HTTP заголовки, параметри і кукі). У розділі «Збір інформації» пояснюється як потрібно проводити тестування під час пасивного етапу.

Методологія тестування безпеки Web-додатків OWASP базується на методі чорного ящика. Інформація про додаток, що тестується, є обмеженою або взагалі відсутня. Тестування поділяють на два етапи пасивний етап та активний.

Під час активного етапу тестувальник проводить тести відповідно до методології, що складається з наступних розділів:

1. Збір інформації.

2. Тестування конфігурації.
3. Тестування політики користувальницької безпеки.
4. Тестування аутентифікації.
5. Тестування авторизації.
6. Тестування управління сесією.
7. Тестування обробки користувальницького введення.
8. Обробка помилок.
9. Криптографія.
10. Тестування бізнес-логіки.
11. Тестування вразливостей на стороні користувача.

Дана методологія має всю необхідну інформації для кожного етапу життєвого циклу безпечної розробки веб-ресурсу. Найбільш популярна і повна методологія для тестування, яка є у відкритому доступі.

Тестування, як один із завершальних етап розробки веб-ресурсу, грає важливу роль в процесі створення якісного програмного забезпечення. При цьому коли справа стосуються безпеки веб-ресурсів, особливо для бізнесу, найкраще рішення буде використовувати вже готові, та популярні методології та стандарти. Для отримання максимального результату їх необхідно застосовувати вже на ранньому етапі розробки. Вибір методології залежить від конкретної ситуації. В якості основи для даної роботи була обрана OWASP Testing Guide.

1.3 Постановка задачі

Розробники веб-сайтів не надають належної уваги питанню безпеки. Існуючі методики іноді потребують специфічних знань і значного часу, тому є складними для звичайного користувача і не кожен зможе перевірити свій веб-додаток на вразливості. Проаналізувавши найбільш поширені вразливості веб-сайтів та існуючі методики тестування безпеки веб-ресурсів, було

прийнято рішення розробити експертну систему для першочергового тестування безпеки веб-сайту і виявлення критичних вразливостей.

Для вирішення завдання розробки експертної системи тестування безпеки веб-ресурсу необхідно виконати наступні завдання:

1. сформувати вхідний математичний опис експертної системи;
2. сформувати базу знань, яка містить опис основних етапів тестування;
3. розробити та програмно реалізувати алгоритми зберігання і обробки таких правил;
4. перевірити працездатність експертної системи.

2 ХАРАКТЕРИСТИКА ЕКСПЕРТНИХ СИСТЕМ ДЛЯ РОЗВ'ЯЗУВАННЯ ЗАДАЧ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

2.1 Основні компоненти експертної системи

Можливість використання експертних систем для вирішення завдань захисту інформації стала цікавити фахівців з інформаційної безпеки в зв'язку з бурхливим розвитком інформаційних технологій, а, отже, і появою нових видів загроз. Уже зараз експертні системи застосовуються для вирішення деяких завдань інформаційної безпеки [10]:

- оцінка ризиків і складання моделі загроз;
- антивірусне програмне забезпечення;
- аудит інформаційної безпеки підприємства.

Незважаючи на все різноманіття завдань, що вирішуються, можна виділити два основні підходи до створення експертних систем:

- створення експертних систем, що використовують евристичні правила;
- створення експертних систем, що самонавчаються.

В експертних системах на основі евристичних правил використовується один з популярних методів представлення знань – правила в формі IF <умова> THEN <action>. Одним із застосувань такого підходу є створення антивірусного програмного забезпечення і систем виявлення вторгнень [11]. Можливі такі варіанти евристичного аналізу:

1) Аналізується програмний код файлу і порівнюється з сигнатурами, що зберігаються в базі антивірусного ПО. Ці сигнатури характеризують не який-небудь конкретний вид шкідливого ПО, а деяку сукупність вірусів, виходячи з припущення про те, що нові віруси мають схожість з вже існуючим шкідливим ПЗ;

2) Аналізуються дії, що здійснюються розглядаються процесом під час роботи, і порівнюються з правилами, збереженими в базі антивірусного ПО. У цьому випадку з'являється можливість виявити шкідливе ПЗ, сигнатури для якого ще не були додані в базу, якщо вона спрямована на виконання тих же дій, що і раніше зустрічалися віруси.

Крім того, евристичні механізми можуть також використовуватися з метою автоматизації аудиту інформаційної безпеки.

Даний підхід до створення експертних систем забезпечує простоту програмування та подання даних, так як знання, що використовуються в розроблюваних системах, можуть бути представлені в порівняно простій формі евристичного правила. Крім того, системи на основі евристичних правил можуть бути розроблені без використання спеціальних засобів (таких, як середовище програмування CLIPS, мова логічного програмування PROLOG). До недоліків подібних систем можна віднести необхідність постійного оновлення баз знань і поліноміальне зростання числа помилкових спрацьовувань системи при надмірній чутливості евристичного аналізатора.

Інший підхід, який можна застосовувати для вирішення задач інформаційної безпеки – використання семантичних мереж. З точки зору математики дана структура є позначений орієнтований граф, вузли якого представляють об'єкти, а дуги – це зв'язки між цими об'єктами [10].

Подібний спосіб представлення знань може бути використаний для опису багатьох предметних областей, в тому числі й належать до сфери інформаційної безпеки. Прикладом використання семантичних мереж для представлення знань в області захисту інформації може слугувати побудова моделі даних про різні вразливості на основі онтологічного підходу, яка потім може бути використана для моделювання мережевих атак.

Розглянемо переваги і недоліки семантичних мереж як способу представлення знань в експертних системах. До переваг можна віднести наступні моменти:

- за допомогою вибору відповідних зв’язків між об'єктами в семантичної мережі, стає можливою опис як завгодно складної предметної області.
- представлена графічно система знань є більш наочною.

Але даний підхід також має і деякі недоліки:

- мережева модель не містить чіткого уявлення про структуру предметної

області;

- подібні моделі є пасивними структурами, а тому вимагають спеціальний апарат формального виведення для обробки;
- при здійсненні пошуку вузлів виникає комбінаторний вибух, особливо якщо відповідь на запит є негативним.

Кожен з описаних вище підходів має власні переваги і недоліки. Проте, представляється можливим комбінування цих підходів з метою впорядкування евристичних правил і прискорення класифікації атаки або шкідливого програмного забезпечення.

У даній моделі передбачається наступна структура: в вузлах графа, що представляє семантичну мережу, знаходяться евристичні правила, що дозволяють віднести атаку або шкідливе програмне забезпечення до того чи іншого типу / класу. Дуги в цій моделі представлятимуть відносини, що показують зв'язок між різними правилами. Самі правила повинні розташовуватися на декількох рівнях, причому кожний наступний рівень повинен визначати більш вузький клас загроз.

Таким чином, побудувавши подібну семантичну мережу, що містить в своїх вузлах евристичні правила для класифікації класу атак або погроз, передбачається отримати збільшення швидкості визначення типу загрози, а, отже, збільшення продуктивності системи виявлення вторгнень.

Щоб експертна система (ЕС) могла отримувати і ефективно використовувати евристичні знання, вони повинні бути представлені в легкодоступному форматі (у вигляді даних, знань і структур управління) [12]. У зв'язку з цим в структурі ЕС виділяються три основні компоненти, які зображенні на рисунку 2.1:

1) БЗ – ядро ЕС, що містить правила і процедури, які використовуються при прийнятті рішення;

2) робоча пам'ять, в якій в кожен момент міститься інформація, надана користувачем і використовувана для прийняття рішення по конкретній проблемі;

3) механізм прийняття рішення (МПР) – це загальний механізм управління, який для прийняття рішення застосовує аксіоматичні знання з БЗ до даних в робочій пам'яті. Зберігає в собі список робочих правил. Це відсортований за пріоритетом список правил, які задовільнять фактам з робочої пам’яті.

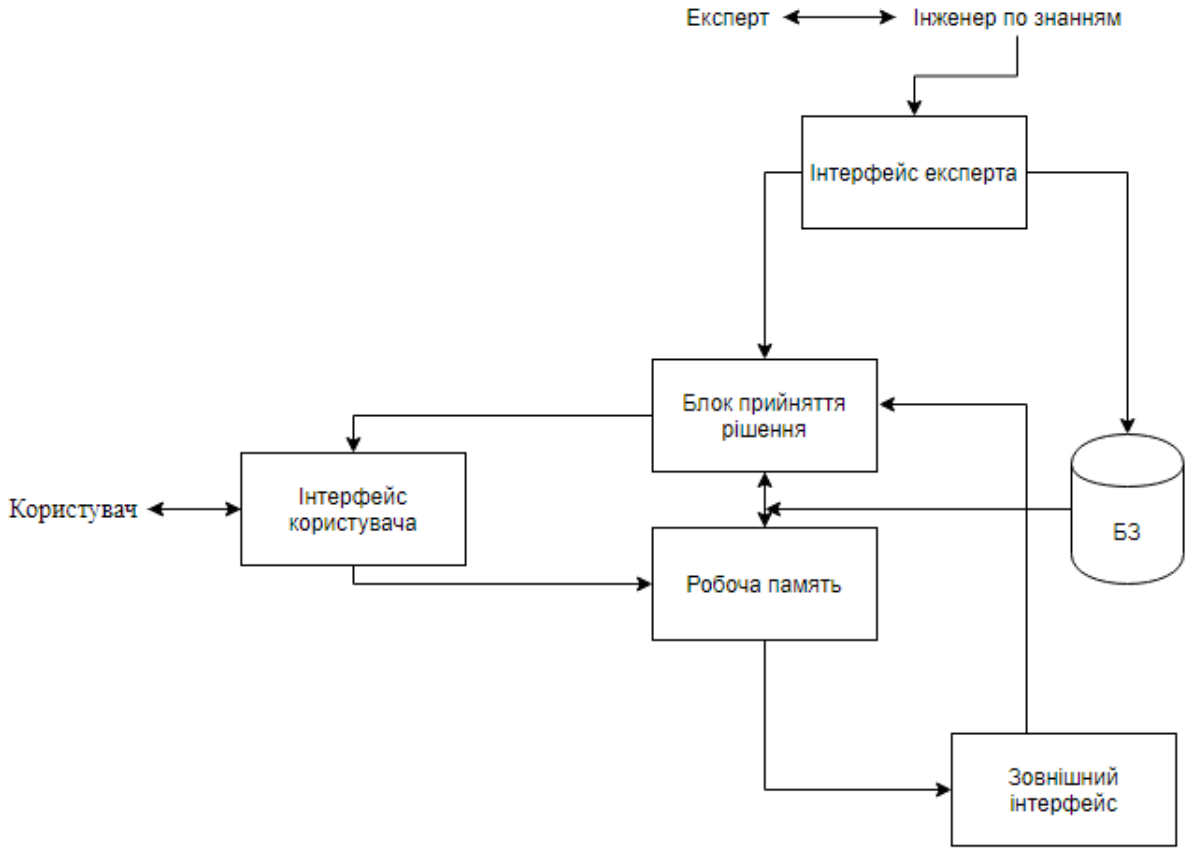

Рисунок 2.1 – Основні компоненти експертної системи

База знань створюється інженерами зі знань. Вони переводять знання експертів (людей) в правила і стратегії, які можуть змінюватися в залежності від сценарію розглянутої проблеми. База знань надає експертній системі можливість давати рекомендації у відповідь на питання користувача і пропонувати йому програму подальших досліджень в області, що є важливою для прийняття рішення, але не очевидною для користувача.

При проведенні аудиту інформаційної безпеки інформаційних систем перед аудиторами постає питання про оцінку захищеності інформаційної системи, а також про вироблення рекомендацій, виконання яких може сприяти підвищенню рівня захищеності. При цьому вартість запропонованих рекомендацій повинна бути не перевищувати вартості інформації, яка захищається, крім того, рекомендації повинні бути максимально ефективними. Очевидно, що при проведенні аудиту необхідно враховувати досвід попередніх аудиторських перевірок. Експертна система аудиту повинна вирішувати наступні завдання:

- автоматизація процедури проведення аудиту;
- полегшення роботи або повна заміна експертів;
- використання накопиченого раніше досвіду;
- вироблення максимально ефективних рекомендацій;
- зниження вартості проведення аудиту.

Інтерфейс користувача призначений для аудиторів або співробітників компанії, що виконують аудит. Через інтерфейс здійснюється первинний вибір загроз, для яких буде проводитися аудит. Через цей же аудитор (співробітник) передає експертній системі необхідні дані, які просить експертна система. Вся інформація, що вводиться користувачем через інтерфейс, передається в робочу пам'ять. Експертна система використовує інтерфейс користувача для надання підсумкових звітів з результатами аудиту та виробленими рекомендаціями. Інтерфейс експерта призначений для передачі знань експертів в базу знань, а також для коригування знань, які вже містяться в ній. Експерт може передавати свої знання системі через інженера по знаннях або самостійно.

2.2 Алгоритми функціонування експертної системи

Як зазначено вище, експертна система складається з бази знань, робочої пам’яті та механізму прийняття рішення. В більшості систем механізм функціонування представлений у вигляді невеликої програми, яка включає два компонента: висновок і управління процесом. Дія першого заснована на

використанні правила «Якщо відомо, що А істина, та існує правило «Якщо А, то В», тоді В істина» [13].

В якості стратегій рішення задач в експертних системах використовуються два типи логічного висновка: прямий логічний висновок і зворотній логічний висновок.

Прямий логічний висновок являє собою метод міркування від факту до висновку, які випливають з цього факту. Наприклад якщо є факт що йде дощ, то з цього випливає, що потрібно взяти зонт.

Зворотній логічний висновок являє собою міркування в зворотному напрямку, від гіпотези (потенційного висновка, яке вже було доведене), до фактів, які підтвердять цю гіпотезу. Наприклад якщо хтось зайшов в дім зі вологим взуттям та зонтом, то можна прийняти гіпотезу що йде дощ. Щоб її підтвердити потрібно запитати в людину чи йде дощ. Якщо відповідь позитивна то гіпотеза вірна, і вона стає фактом. Як було сказано вище, гіпотеза може бути розглянута як факт, істина якого викликає сумніви і повинна бути встановлена. В такому випадку гіпотеза інтерпретуєтеся, як ціль яка повинна бути доведення [12, 13].

В залежності від експертної системи в механізмі прийняття рішення реалізовується як прямий або зворотний логічний вибір, так і одночасно дві форми логічного вибору.

В системах, база знань яких досить велика, потрібно мати стратегію управління вибору рішення, яка дозволяє мінімізувати час пошуку рішення. До таких стратегій відноситься пошук в ширину, пошук в глибину.

При пошуку в глибину в якості чергової підцілі вибираєтеся та, яка відповідає наступному, більш детальному рівню опису задачі. Наприклад якщо є симптоми, які можуть діагностувати захворювання, потрібно шукати уточнювальні симптоми поки не спростується дана гіпотеза.

При пошуку в ширину, навпаки, система спочатку аналізує всі симптоми які знаходяться на одному рівні, навіть якщо вони відносяться до різних захворювань і тільки потім відбувається перехід до наступного рівня (рис. 2.2)

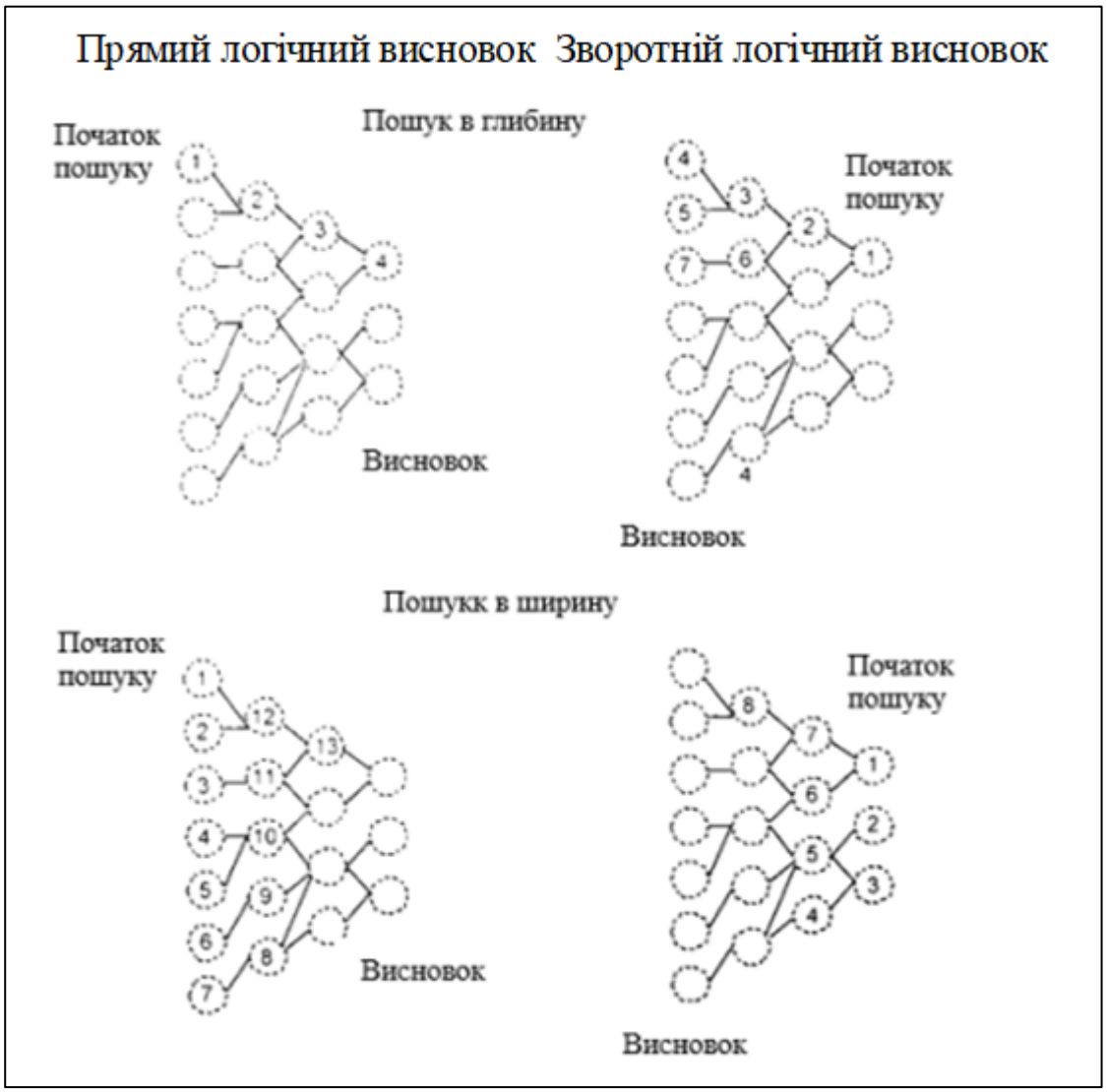

Рисунок 2.2 – Схема пошуку в глибину та пошуку в ширину, для прямого логічного висновка та зворотного логічного висновка

Розглянемо механізм роботи експертної системи на прикладі вибору дії під час переходу дороги зі світлофором. Цю систему можна виразити наступним псевдокодом:

Правило red:

IF: горить червоне світло

THEN: стояти

Правило green:

IF: горить зелене світло

THEN: йти

Кожне правило має ім’я (red, green), частина правила після «IF», називається умовною частиною (антецедент), кожна окрема взята умова називається шаблоном.

Робоча пам'ять може включати в собі факти, які стосуються поточного стану. В робочій пам’яті можуть бути будь-який з фактів або декілька одночасно. Якщо світлофор працює нормально то в робочій пам’яті буде знаходитися тільки один факт. Але можливе і таке, що в робочій пам’яті будуть одразу обидва факти, з чого може випливати що світлофор несправний. Потрібно звернути уважно, но те в чому різниця, між базою знань та робочою пам’яттю. Факти не взаємодіють один з одному. Факт «горить зелене світло» не впливає на факт «горить червоне світло». З іншого боку, наші знання про світлофор говорять про те, якщо зустрічаються одночасно ці два факти, це означає, що світлофор несправний.

Якщо в робочій пам’яті є факт що «горить зелене світло», то механізм прийняття рішення розуміє, що цей факт задовольняє умовну частину правила «green» і поміщає це правило в робочий список правил. А якщо правило має декілька шаблонів то всі шаблони мають задовольняти одночасно і тільки тоді правило можна буде помістити в робочий список правил. В якості умови деяких шаблоні можна вказати відсутність зазначених фактів в робочій пам’яті.

Правило в якого всі шаблони задовільні називається активізованими або реалізованими. В робочому списку правил може буду одночасно декілька таких правил. В цьому випадку механізм прийняття рішення має вибрати одне із правил для запуску [14].

Після умови в правилі знаходиться список дій (консеквентна). Якщо виконується запуск правила виконується його дія. В склад конкретної дій зазвичай входить додати або видалити факт із робочої пам’яті або виведення

результату. Формат написання таких дій залежить від синтаксиста мови експертної системи.

Машина логічного вибору рішення працює в режимі виконування циклів «розпізнавання – дія» . Механізм прийняття рішення знову і знову виконує деяку групу задач до виявлення визначених критеріїв, які викликають завершення роботи програми. При цьому вирішуються деякі спільні задачі, такі як вирішення конфліктів, дія, узгодження, перевірка умов, завершення (рис. 2.3).

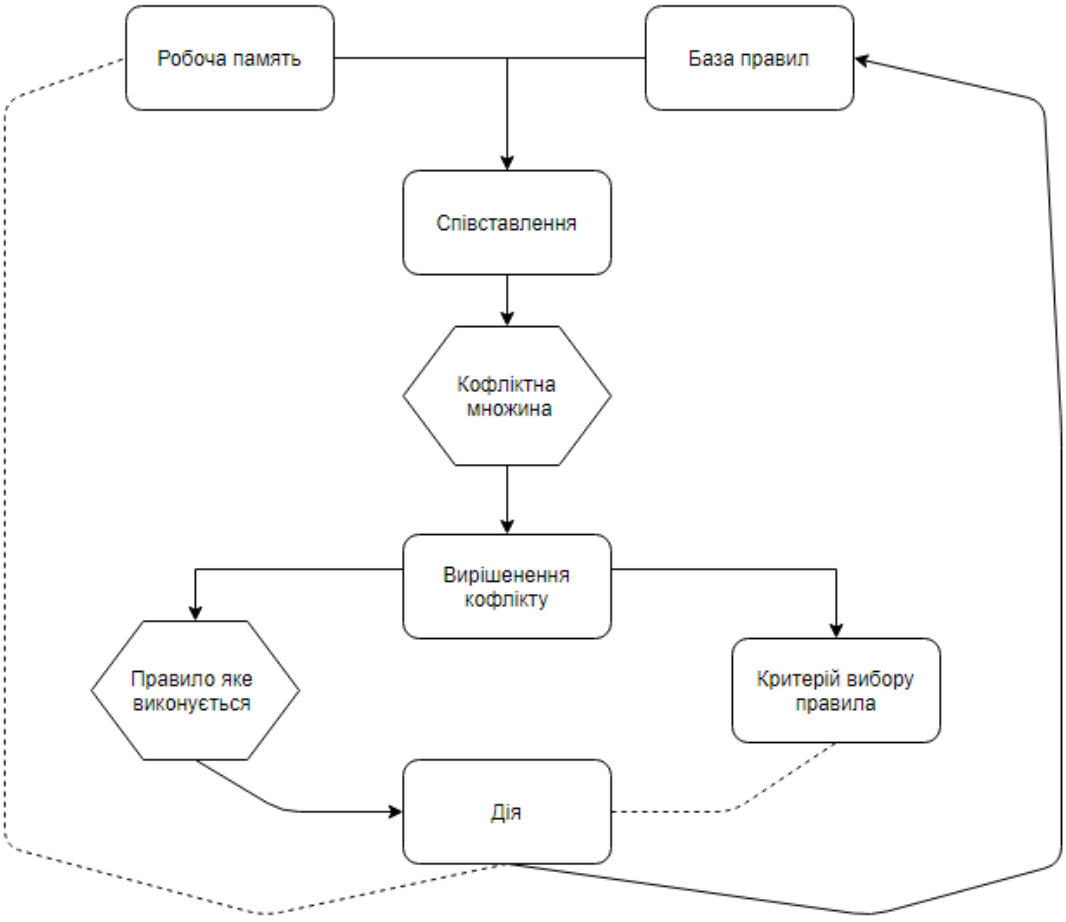

Рисунок 2.3 – Схема роботи машини логічного вибору

Протягом кожного циклу може бути активізовані та переміщені в робочий список безліч правил. Крім того, в робочому списку правил залишаються результати активізацій від попередніх циклів, якщо не виконується деактивація цих правил в зв’язку с тим, що їх ліва частина більше не виконується. Таким чином в ході виконування програми кількість

активізованих правил в робочому списку правил змінюється. В залежності від програми раніше активізовані правила можуть бути назавжди залишатися в робочому списку правил, але ніколи не вибиратися для запуску. Аналогічним чином деякі правила можуть ніколи не бути активізованими. В таких випадках необхідно повторно перевірити визначання цих правил, оскільки або ці правила не потрібні, або шаблони неправильно спроектовані.

Механізм прийняття рішення виконує дію активізованого правила з найвищим пріоритетом з робочого списку правил, потім активізованого правило зі наступним по порядку пріоритетом і так далі , до поки, в списку більше не залишиться більше активізованого правила. Для інструментальних експертних систем розроблено багато різних систем пріоритетів, але в загалом всі інструментальні інструменти дозволяють інженеру по знанням встановлювати пріоритети правил [15].

В робочому списку виникають конфлікти якщо різні активізовані правила мають однаковий пріоритет і механізм прийняття рішення повинен прийняти рішення, яке з цих правил необхідно запустити.

Після завершення виконання всіх правил, управління повертається до інтерпретатора команд верхнього рівня, щоб користувач міг надати додаткові інструкції.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЕКСПЕРТНОЇ СИСТЕМИ ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-РЕСУРСІВ

3.1 Програмні засоби для проектування експертних систем

Розробку експертної системи можна умовно поділити на 6 етапів [16].

Етап 1. Ідентифікація.

Описуються задачі, які потрібно виконати, плануються етапи розробки прототипів експертної системи, визначають: найсучасніші ресурси (книги, додаткові спеціалісти, методики), що містять аналогічні аналітичні системи, цілі, класи задач тощо.

Інженер, який знає дану область, допомагає експертам виявити та створити знання, необхідні для роботи експертної системи, з використанням різних способів: аналіз текстів, діалогів, експертних ігор, дискусій, інтерв'ю, спостереження та інших. Вивільнення знань – це надання інженером за знанням більшого повного представлення про предмету область сфери та методів прийняття рішень.

Етап 2. Концептуалізація.

Створюється структура отриманих знань про предметну область. Визначаються: термінологія, переклад головних понять та їх атрибутів, структура вхідної та вихідної інформації, стратегія прийняття рішення та ін. Концептуалізація - це розробка неформального опису знань про предметну область у вигляді графа, таблиць, діаграм або будь-яких текстів, які охарактеризують основні концепції та взаємозв'язки між поняттями предметної області.

Етап 3. Формалізація.

На етапі формалізації всіх ключових понять і відносин, виявлених на етапі концептуалізації, виражаються формальній мові, запропонованому інженером за знанням. Тут вирішується, які інструментальні засоби використовуються для вирішення проблем, що розглядаються.

Етап 4. Реалізація.

Створюється прототип експертної системи, що включає базу знань та інших підсистем. На даному етапі застосовуються наступні інструментальні засоби: програмування на звичайних мовах (Паскаль, Сі та ін.), програмування на спеціалізованих мовах, присутніх у задачах інтелекту (LISP, FRL, SmallTalk та ін.) та інші. Четвертий етап в розробці експертні системи в якомусь ступені є ключовим, так як на ньому відбувається створення програмного комплексу.

Етап 5. Тестування.

Прототип перевіряється на зручність та адекватність інтерфейсів введення-виведення, ефективність стратегії управління, корекція бази знань. Тестування – це виявлення помилок у вибраному підході, виявлення помилок у здійсненні прототипу, а також випробовування рекомендацій за введеною системою до промислового варіанту.

Етап 6. Експлуатація.

Перевіряє працездатність експертної системи на кінцевих користувачах. За результатами цього етапу можна використовувати необхідну модифікацію експертної системи.

Важливо в розробці системи правильно вибрати інструменти розробки. На даний момент найпопулярніші мови програмування для створення експертних систем є LISP та PROLOG. Серед програмних середовищ, для розробки можна виділити KEE, CENTAUR, G2 і GDA, CLIPS, АТ_ТЕХНОЛОГІЯ. Вони зручні тим, що дають можливість інженеру зі знань, без великих навиків програмування, використовувати комбіновані системи представлення знань, зручну роботу з об’єктами та процедурами [15].

При виборі інструменту важливо розуміти, який тип експертної системи планується розробити. За функцією експертні системи можна поділити на: загального призначення, вузьконаправлені системи, для рішення проблем з діагностики або проектування, для рішення проблем специфічних завдань.

ЕС CLIPS була обрана як інструментальний засіб для розробки. Вибір пояснюється тим, що в CLIPS вбудована об'єктно-орієнтована мова програмування COOL, тому за допомогою CLIPS можна створювати ЕС будь-якої складності. Крім того, CLIPS дозволяє використовувати продукційні правила типу «Якщо - То», за допомогою яких зручно представляти невеликі фрагменти знань, а також реалізовувати логічний висновок. Виходить, що CLIPS поєднує в собі переваги двох категорій інструментальних засобів ЕС: заснованих на правилах і на об'єктах. Серед інших переваг оболонки CLIPS можна виділити наступні:

- безкоштовне поширення системи;
- наявність великої документації разом з простотою мови, що прискорюють процес навчання написання програм на CLIPS;
- висока швидкість роботи створених на CLIPS додатків;
- можливість легко створювати, експлуатувати і супроводжувати на CLIPS великі БЗ, використовуючи об'єкти. Крім того, для CLIPS на мові С ++ можна створювати розширення, а також інтегрувати додатки CLIPS в програми.

3.2 Формування бази знань

При формуванні бази знань потрібно враховувати, що тестування веб-ресурсу – є процесом унікальним для кожного випадку оскільки для різних ресурсів необхідний різний рівень безпеки, також відрізняється кількість потенційних вразливостей та частин для тестування. Спочатку необхідно визначити тестування, які необхідно добавити в план аудиту. Також необхідно врахувати, що користуватися буде спеціаліст, який має уявлення про роботу веб-ресурсу та вміє користуватися інструментами та засобами, які будуть запропоновані в ході використання експертної системи, тому його введені дані будуть коректі та правдиві. Головною метою розроблюваної системи є надання інформаційної консультації з питань які частини необхідно

протестувати, щоб впевнитися в безпеці веб-ресурсу та зробити оцінку ризику кожної потенційної загрози.

Спочатку необхідно з’ясувати, які тести необхідно провести. Під час первинного тестування необхідно пройти всі тести.

1. Якщо не відбувався збір інформації – провести необхідні тести.
2. Якщо не було тестування конфігурації та розгортання – провести необхідні тести.
3. Якщо не було тестування управління ідентичністю – провести необхідні тести.
4. Якщо не було тестування автентичності – провести необхідні тести.
5. Якщо не було тестування авторизації – провести необхідні тести.
6. Якщо не було тестування сеансу управління – провести необхідні тести.
7. Якщо не було тестування перевірки даних – провести необхідні тести.
8. Якщо не було тестування помилок обробки – провести необхідні тести.
9. Якщо не було тестування криптографії – провести необхідні тести.
10. Якщо не було тестування ділової логіки – провести необхідні тести.
11. Якщо не було тестування на стороні клієнта – провести необхідні тести.

Опис тестів та необхідні рекомендації щодо їх виконання наведено в таблиці 3.1 [10].

Таблиця 3.1 – Опис тестів.

Ім’я тесту	**Опис тесту**	**Інструмент**
Тестування збору інформації		
Проведення пошуку та розвідки пошукової системи для пошуку витоку інформації	Використовуйте пошукову систему для пошуку мережевих діаграм та конфігурацій, облікових даних, вмісту повідомлення про помилку.	Google Hacking, Sitedigger, Shodan, FOCA, Punkspider
Відбиток веб-сервера	Знайдіть версію та тип запущеного веб-сервера, щоб визначити відомі вразливості та відповідні подвиги	Httprint, Httprecon, Desenmascarame

Продовження табл. 3.1

Ім’я тесту	Опис тесту	Інструмент
Перегляньте метафайли веб-сервера на предмет витоку інформації	Проаналізуйте robots.txt та визначте <META> теги з веб-сайту.	Browser, curl, wget
Перерахуйте програми на веб-сервері	Знайдіть програми, розміщені на веб-сервері (віртуальні хости / субдомен), нестандартні порти, передачі зони DNS	Webhosting.info , dnsrecon, Nmap, fierce, Recon-ng, Intrigue
Перегляньте коментарі та метадані на веб-сторінці щодо витоку інформації	Знайдіть конфіденційну інформацію з коментарів веб-сторінок та метаданих про вихідний код.	Browser, curl, wget
Тестування конфігурації та розгортання		
Тестування обробки розширень файлів для чутливої інформації	Пошук важливих файлів, інформацію (.asa, .inc, .sql, zip, tar, pdf, txt тощо)	Nessus
Файли резервного копіювання та невикористання для конфіденційної інформації	Перевірте вихідний код JS, коментарі, файл кешу, файл резервної копії (.old, .bak, .inc, .src) та відгадайте ім'я файлу	Nessus, Nikto, Wikto
Перерахування інфраструктури та інтерфейсу адміністратора додатків	Перерахуйте каталоги та файли, коментари та посилання у джерелі (/ admin, / administrator, / backoffice, / backend тощо), альтернативний порт сервера (Tomcat / 8080)	Burp Proxy, dirb, Dirbuster, fuzzdb, Tilde Scanner
Тестування методів HTTP	Ідентифікуйте дозволені методи HTTP на веб-сервері за допомогою OPTIONS. Довільні методи HTTP, байпас управління доступом HEAD та XST	netcat, curl
Сувора безпека транспорту HTTP	Визначте заголовок HSTS на веб-сервері за допомогою заголовка відповіді HTTP. завиток -s -D- https://domain.com/ | | grep Strict	Burp Proxy, ZAP, curl
Перегляньте метафайли веб-сервера на предмет витоку інформації	Проаналізуйте robots.txt та визначте <META> теги з веб-сайту.	Browser, curl, wget

Продовження табл. 3.1

Ім’я тесту	Опис тесту	Інструмент
Тестування перевірки даних		
Тестування XSS	Перевірте перевірку на вхід, замініть вектор використовуваний для ідентифікації XSS, на XSS з забрудненням параметра HTTP.	Burp Proxy, ZAP, Xenotix XSS
Тестування на SQL-ін'єкцію	Union, Boolean, Error based, Out-of-band, Time delay.	Burp Proxy (SQLipy), SQLMap, Pangolin, Seclists (FuzzDB)
Тестування Oracle	Визначте URL-адреси для веб-застосунків PL / SQL, Доступ із пакетами PL / SQL, Обхід списку виключень PL / SQL, Введення SQL	Orascan, SQLInjector
Тестування MySQL	Визначте версію MySQL, Єдину цитату, Інформаційну схему, Файл читання / запису.	SQLMap, Mysqloit, Power Injector
Тестування доступу MS	Перерахуйте стовпчик на error-based (Group by), Отримати схему бази даних поєднати з fuzzdb.	SQLMap
Тестування на ін'єкцію NoSQL	Визначте бази даних NoSQL, передайте спеціальні символи ('"\; {}), Атака із зарезервованою назвою змінної, оператором.	NoSQLMap
Тестування на ін'єкцію LDAP	"/ldapsearch?user=* user=*user=*)(uid=*))(|(uid=*pass=password"	Burp Proxy, ZAP
Тестування на ін'єкцію XML	Перевірте XML мета символи ', "", <>, <! - / ->, &, <! [CDATA [/]]>, XXE, TAG	Burp Proxy, ZAP, Wfuzz

Продовження табл. 3.1

Ім’я тесту	Опис тесту	Інструмент
Тестування криптографії сайту		
Тестування на Padding Oracle	Перевірте данні трьох станів: • Шифровий текст розшифровується, а отримані дані є правильними. • Шифровий текст розшифровується, отримані дані є неправильними але викликаються деякий обробник виняткових подій або механізм роботи з помилками в логіці програми. • Розшифровка тексту шифру не вдається через помилки заміщення. "	PadBuster, Poracle, python-paddingoracle, POET
Тестування на чутливу інформацію, що надсилається через незашифровані канали	Перевірте конфіденційні дані під час передачі: • Інформація, що використовується для автентифікації (наприклад, уповноважені дані, PIN-коди, Сесія ідентифікатори, маркери, файли cookie ...) • Інформація, захищена законами, нормативними актами або конкретними організаційними політика (наприклад, кредитні картки, дані клієнтів)	Burp Proxy, ZAP, Curl
Тестування бізнес логіки		
Тестування перевірки бізнес-логіки	Пошук точок введення даних або передачі точок між системами або програмним забезпеченням. • Після виявлення спробуйте вставити логічно недійсні дані в додаток / систему.	Burp Proxy, ZAP

Продовження табл. 3.1

Ім’я тесту	Опис тесту	Інструмент
Тест на здатність формувати запити	Пошук передбачувану або приховану функціональність полів. • Після виявлення спробуйте вставити логічно дійсні дані у додаток / систему, що дозволяє користувачеві пройти додаток / систему проти нормального робочого процесу логіки бізнесменів.	Burp Proxy, ZAP
Перевірка цілісності тесту	Пошук частин програми / системи (компонентів, наприклад, поля введення, бази даних або журнали), які переміщують, зберігають або обробляють дані / інформацію. • Для кожного визначеного компонента визначте, який тип даних / інформації є логічно прийнятним та які типи додатків / системи повинні захищати. Також врахуйте, кому відповідно до бізнес-логіки дозволено вставляти, оновлювати та видаляти дані / інформацію та в кожному компоненті. • Спроба вставити, оновити або відредагувати видалити значення даних / інформації з недійсними даними / інформацією в кожен компонент (тобто вхід, базу даних або журнал) користувачами, які не повинні бути дозволені за робочим процесом логіки бізнесу	Burp Proxy, ZAP
Тест на терміни процесу	• Шукаєте функціональні можливості додатків / систем, які можуть впливати часом. Наприклад, час виконання або такі дії допомогти користувачам передбачити майбутній результат або дозволити його обійти будь-яка частина бізнес-логіки чи робочого процесу. Наприклад, ні завершення транзакцій у очікуваний час. • Розробити та виконувати випадки неправомірного використання, забезпечуючи зловмисників не може отримати перевагу на основі будь-якого часу ».	Burp Proxy, ZAP
Тестування на перевищення кількості виклику функцій	• Пошук функцій або функцій у додатку чи системі, які не повинні виконуватись більше, ніж один раз або задану кількість разів протягом робочого процесу бізнес-логіки. • Для кожної з виявлених функцій та функцій, які слід виконувати лише одноразово або вказану кількість разів протягом робочого процесу бізнес-логіки, розробляйте випадки зловживань.	Burp Proxy, ZAP

Продовження табл. 3.1

Ім’я тесту	Опис тесту	Інструмент
Тестування на стороні клієнта		
Тестування сценарію крос-сайтів на основі DOM	Тест на введення даних користувачів, отриманих від клієнтських об’єктів JavaScript	Burp Proxy, ZAP
Тестування на ін'єкцію HTML	Надіслати шкідливий HTML-код: ? user = <img% 20src = 'aaa'% 20onerror = попередження (1)>	Burp Proxy, ZAP
Тестування на перенаправлення на сторону клієнта	Змінити ненадійне введення URL-адреси на шкідливий сайт: (Відкрити перенаправлення) ? redirect = www.fake-target.site	Burp Proxy, ZAP
Тестування на ін'єкцію CSS	Код введення в контекст CSS: • www.victim.com/#red;-o-link:'javascript:alert(1)';-o-link-source:current; (Opera [8,12]) • www.victim.com/#red ;-:expression(alert(URL=1)); (IE 7/8)	Burp Proxy, ZAP
Тестування на маніпуляцію на стороні клієнта	Зовнішній JavaScript можна легко вставити на надійний веб-сайт www.victim.com/#http://evil.com/js.js	Burp Proxy, ZAP
Тестування на Clickjacking	Виявіть, чи веб-сайт вразливий, завантажившись у рамку iframe, створіть просту веб-сторінку, що включає рамку, що містить ціль.	Burp Proxy, ClickjackingTool
Тестування WebSockets	Визначте, що програма використовує WebSockets, перевіривши схему ws: // або wss: // URI. Перевірте походження, конфіденційність та цілісність, автентифікацію, авторизацію, санітарію введення	Burp Proxy, Chrome, ZAP, WebSocket Client

В якості відповіді після проведення кожного тесту, система має отримати результат «Pass», «Issue», «NA». Це, в свою чергу, додає в базу факт типу (<назва тесту> <результат>). Якщо після перевірки певний тест має результат «Issue», то виконується оцінка ризиків даної загрози.

Для цього виконується згідно відповідей аудитора оцінка за двома факторами: імовірності виконання та вплив на систему.

Для оцінки імовірності використовується фактори загрози (рівень, мотив, можливість, виконавці) та фактори вразливості (знаходження, реалізація, інформованість, виявлення вторгнення). Для оцінки впливу на систему розглядаються фактори технічного впливу (втрата конфіденційності, цілісності, доступності, звітності) та фактори впливу на бізнес (фінансовий, репутаційний, невідповідність, політика конфіденційності). Кожен фактор має свої варіанти відповідей, які мають певне числове значення необхідні для розрахунку (табл. 3.1 – 3.5). Оцінка отримується шляхом знаходження середньо арифметичного числових значень факторів .

Таблиця 3.2 – Фактори загрози

Фактори	**Можливі оцінки та її числові значення**	
Необхідні навички	Не застосовується	0
	Немає технічних навичок	1
	Деякі технічні навички	3
	Просунутий користувач комп’ютера	5
	Навички мережі та програмування	6
	Навички проникнення безпеки	9
Мотив	Не застосовується	0
	Низька або без винагороди	1
	Можлива винагорода	4
	Висока нагорода	2
Можливість	Повний доступ або необхідні дорогі ресурси	0
	Необхідний спеціальний доступ або ресурси	4
	Необхідний певний доступ або ресурси	7
	Не потрібен доступ чи ресурси	9

Виконавці	Не застосовується	0
	Системні адміністратори	2
	Користувачі внутрішньої мережі	4
	Партнери	5
	Автентифіковані користувачі	6
	Анонімні користувачі Інтернету	9

Таблиця 3.3 – Фактори вразливості

Знаходження	Не застосовується	0
	Практично неможливо	1
	Складно	3
	Легко	5
	Доступні автоматизовані інструменти	9
Реалізація	Не застосовується	0
	Теоретично	1
	Складно	3
	Легко	7
	Доступні автоматизовані інструменти	9
Інформованість	Не застосовується	0
	Невідомі знання	1
	Приховано знання	4
	Очевидні знання	6
	Загальнодоступні знання	9
Виявлення	Не застосовується	0
	Активне виявлення в додатку	1
	Зареєстровано та переглянуто	3
	Увійшли без огляду	8
	Не зареєстровано	9

Таблиця 3.4 – Фактори технічного впливу

Фактори	**Можливі оцінки та її числові значення**	
Втрата кондефіційності	Не застосовується	0
	Розкриті мінімальні нечутливі дані	2
	Розкриті широкі нечутливі дані	6
	Розкриті широкі критичні дані	7
	Усі розкриті дані	9
Втрата цілісності	Не застосовується	0
	Мінімально злегка пошкоджені дані	1
	Мінімально серйозно корумповані дані	3
	Обширні незначні корумповані дані	5
	Значні з корумповані дані	7
	Усі дані повністю пошкоджені	9
Втрата доступності	Не застосовується	0
	Мінімальні вторинні послуги перервано	1
	Мінімальні первинні послуги перервано	5
	Перервано широкі первинні послуги	7
	Усі послуги повністю втрачені	9
Втрата звітності	Не застосовується	0
	Атака повністю простежується до особи	1
	Атака, можливо, простежується до особи	7
	Атака абсолютно анонімна	9

Таблиця 3.5 – Фактори впливу на бізнес

Фактори	**Можливі оцінки та її числові значення**	
Фінансовий	Не застосовується	0
	Пошкодження шкоди менше, ніж для вирішення проблеми	1
	Незначний вплив на річний прибуток	3
	Значний вплив на річний прибуток	7
	Банкрутство	9
Репутаційний	Не застосовується	0
	Мінімальний збиток	1
	Втрата основних рахунків	4
	Втрата доброї волі	6
	Пошкодження торгової марки	9
Невідповідність	Не застосовується	0
	Незначне порушення	2
	Явне порушення	5
	Порушення високого профілю	7
Політика кондефіційності	Не застосовується	0
	Одна особа	3
	Сотні людей	5
	Тисячі людей	7
	Мільйони людей	9

Для оцінки впливу на систему розглядаються фактори технічного впливу (втрата конфіденційності, цілісності, доступності, звітності) та фактори впливу на бізнес (фінансовий, репутаційний, невідповідність, політика конфіденційності).

Для отримання загальної оцінки ризику використовуються наступні правила:

1) Якщо оцінка імовірності менше 3 та оцінка впливу менше 3, оцінка – немає ризику.
2) Якщо оцінка імовірності менше 3 та оцінка впливу більше 3, але менше 6, оцінка – низький рівень ризику.
3) Якщо оцінка імовірності більше 3, але менше 6, та оцінка впливу менше 3, оцінка – низький рівень ризику.
4) Якщо оцінка імовірності менше 3 та оцінка впливу більше 6, оцінка – помірний рівень ризику.
5) Якщо оцінка імовірності більше 6 та оцінка впливу менше 3, оцінка – помірний рівень ризику.
6) Якщо оцінка імовірності більше 6 та оцінка впливу більше 3, оцінка – високий рівень ризику.
7) Якщо оцінка імовірності більше 3 та оцінка впливу більше 6, оцінка – високий рівень ризику.

Результатом буде база фактів типу (risk score <name_test> <number>)

3.3 Програмна реалізація експертної системи

Експертну систему було реалізовано за допомогою мови програмування LISP, та середовища CLIPS у вигляді проекту audit.clp.

Таблиця 3.6 – Основні функції audit.clp.

№	Назва функції	Короткий опис
1	(deffunction ask-que (?q $?all-values)	Задає користувачу питання ?q, та повертає відповідь від користувача. Відповідь має бути в спису &?all-values
2	(deffunction y-or-n-q (?q))	Задає користувачу питання. Повертає відповідь yes або no.

№	Назва функції	Короткий опис
	(deffunction Pass-or-Issue-or-NA-test (?test)	Задає користувачу питання ?test. В якості відповіді приймає Pass, NA, Issue. Створює факт типу (?test (відповідь)).
3	(deffunction make-test (?name-test))	Функція для оцінки ризику. Створює факти оцінку імовірності, та оцінку впливу.

В розробленій експертній системі було реалізовано функцію, для задавання необхідних питання та отримання на них відповідей у певній заданій формі. Оскільки всі питання, та варіанти відповіді були описані, в строгій формі в попередньому пункті.

Правило which_test_need, яке активується після запуску програми, визначає, які види тестування необхідно зробити, та добавляє їх в базу фактів. Додатково створює факт (start test).

У експертній системі реалізуються правила, які будуть виводити на екран необхідні данні щодо проведення тестування, та задаватись певні запитання, щодо результату проходження тесту. При необхідності буде запускатися функція (deffunction make-test (?name-test), яка через запитання, дасть оцінку ризику тесту який має вплив на систему. Для обчислення оцінки ризику було створено правило find_score, яке активується, якщо в базі наявні факти оцінки імовірності, та оцінки впливу, до конкретного етапу тестування.

Для початку роботи з експертною системою, необхідно запустити команду reset, добавить всі необхідні факти для початку роботи, та запустити командою run. Одразу появиться повідомлення "Експертна система тестування веб-ресурсу". Після цього користувач буде отримувати рекомендації та запитання необхідних для оцінки та проведення тестування.

Повний текст програми наведено у додатку.

3.4 Тестування експертної системи

Працездатність розробленої експертної системи перевіряється шляхом виконання експертного аудиту на визначення критичних частин безпеки веб-ресурсу.

Припустимо ситуацію, що під час розробки додатку особистого кабінету для університету розробник допустив помилку, та зловмисник потенційно через SQL-ін’єкцію зможе отримати особисті дані студентів та співробітників. Перш ніж запустити додаток для користування виконується аудит за допомогою розробленої системи.

На початку роботи система запитує які типи тестування необхідно додати в план аудиту. Виконання цього етапу зображено на рисунку 3.5.

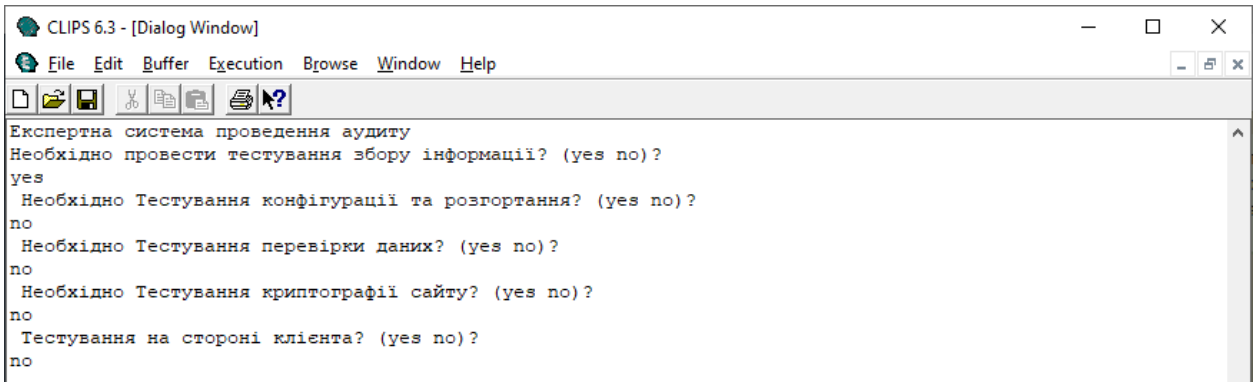

Рисунок 3.1 – Вибір плану аудиту

На наступному етапі система дає рекомендації щодо інструментів, які необхідно використовувати для проведення тестів. Аудитор вписує результат тестів, як зображено на рисунку 3.2.

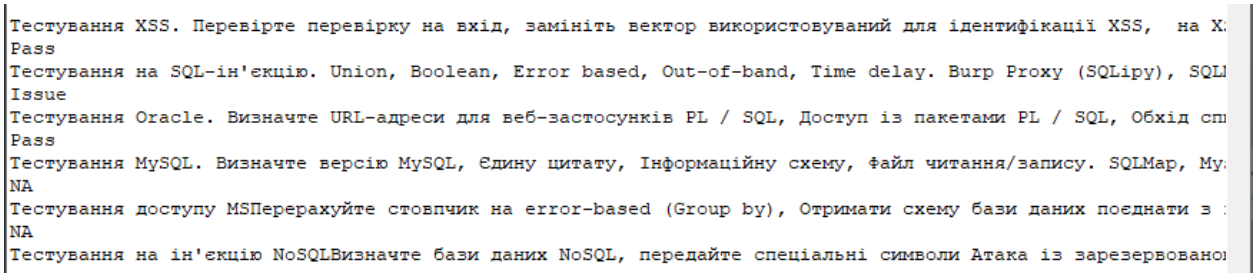

Рисунок 3.2 – Введення результатів тестів

Після проходження всіх запланованих тестів система виконує оцінку ризиків для частин аудиту, які не відповідають вимогам безпеки (рис. 3.3). Для

цього використовується методика для оцінки ризику вразливостей, яка описана у розділі 3.2.

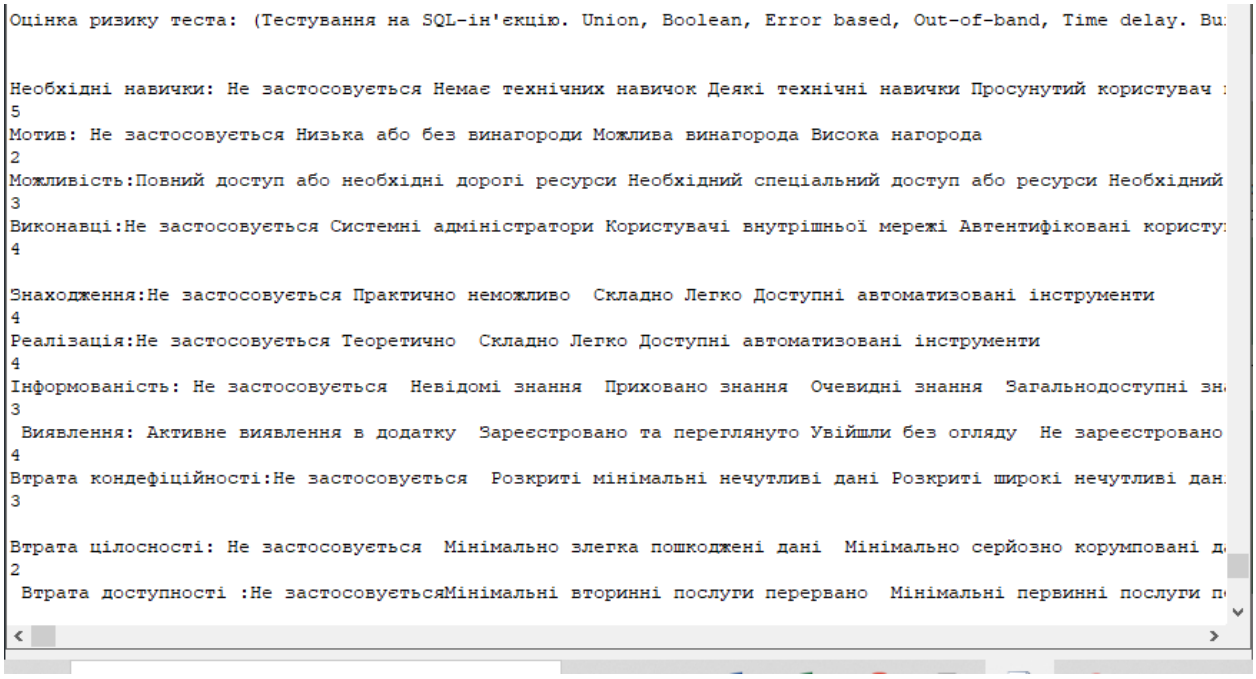

Рисунок 3.3 – Проведення оцінку ризиків

У результаті аудитор отримує базу фактів, з якої можна зробити висновок про існуючі вразливості та їх критичність. Також експертна система виводить на екран оцінку ризиків. Як можна побачити на рис. 3.4, що вразливість SQL-ін’єкції має високий рівень ризику. Отже, на основі отриманого результату, аудитор може зробити висновок, що запуск додатку необхідно відкласти, поки ця проблема не вирішиться. Далі можна надати звіт розробнику.

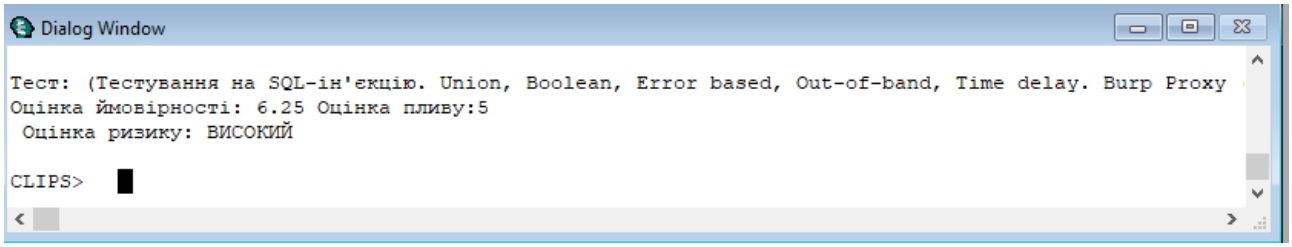

Рисунок 3.4 – Результат оцінювання ризику

ВИСНОВКИ

Дана робота присвячена виявленню вразливостей в додатках, що виникли через недоліки забезпечення безпеки. Рішення даної проблеми можливо з тестуванням безпеки web-додатка за допомогою методології OWASP testing guide. Чим складніше сайт, тим більше часу потрібно на його перевірку і налагодження. На жаль, існує безліч прикладів, коли розробники і замовники втрачають етап тестування сайту, що практично завжди призводить до великих фінансових і тимчасових витратах в подальшому, невдоволення користувачів ресурсу і в результаті, необхідність доопрацювання (або навіть повторної розробки) ресурсу. Але провівши повний цикл тестування безпеки, не можна бути на 100% впевненим, що система по-справжньому безпечна. Але можна бути впевненим в тому, що відсоток несанкціонованих проникнень, крадіжок інформації і втрат даних буде в рази менше, ніж у тих, хто не проводить тестування безпеки.

У випускній роботі була спроектована і реалізована експертна система тестування безпеки веб-ресурсу з використанням продукційної моделі подання знань. Розроблена експертна система може бути використана при первинному та повторних тестуваннях веб-ресурсів. Працездатність експертної системи перевірялася шляхом моделювання можливої ситуації. У подальшому планується розширити базу знань, додавши інші методології та розробити зручний графічний інтерфейс для користувачів.

СПИСОК ЛІТЕРАТУРИ

1. ISO/IEC 29147:2018 Information technology — Security techniques — Vulnerability disclosure / 2018
2. НД ТЗІ 1.1–003–99 Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу - 1999.
3. WASC Threat Classification. WASC [Електроний ресурс] / Robert Auger - 2012 — Режим доступу: http://projects.webappsec.org/w/page/13246978/Threat%20Classification
4. OWASP TOP TEN [Електроний ресурс] / Andrew van der Stock, Brian Glas, Neil Smithline, Torsten Gigler - 2017— Режим доступу: https://owasp.org/www-project-top-ten/
5. Скабцов Н. Аудит безопасности информационных систем / Н. Скабцов. – СПб.: Питер, 2018. – 272 с.
6. Тецкий А.Г. Методы и средства тестирования на проникновение веб-приложений и сетей. Практикум / под ред. В.С. Харченко – НАУ им. Н.Е. Жуковского «ХАИ». 2017. – 77 с.
7. Open Source Security Testing Methodology Manual (OSSTMM) : Режим доступу: https://www.isc2.org/ .
8. Penetration Testing Execution Standard (PTES) [Електроний ресурс] / [Chris Nickerson, Dave Kennedy, та ін.] - 2014 — Режим доступу: http://www.pentest-standard.org/index.php/Main_Page
9. OWASP Testing Guide [Електроний ресурс] / Elie Saad, Rick Mitchell, Matteo Meucc — Режим доступу: https://owasp.org/www-project-web-security-testing-guide/
10. Болтунов А.И. Применение экспертных систем для решения задач информационной безопасности / А.И. Болтунов, Л.Н. Кротов // Международный научно-исследовательский журнал. – 2016. – № 9(51), Часть 2. – С. 9-12.
11. Частиков А. П. Разработка экспертных систем. Среда CLIPS /

А. П. Частиков, Д. Л. Белов, Т. А. Гаврилова. – СПб.: BHV, 2003. –641 с.

12. Джексон П. Введение в экспертные системы. – Вильямс, 2001. – 624 с.
13. Гаврилова Т. А. Базы знаний интеллектуальных систем. Учебник. / Т.А. Гаврилова, В.Ф. Хорошевский. – СПб.: Питер, 2000.
14. Джозеф Джарратано, Гари Райли. «Экспертные системы: принципы разработки и программирование» = Expert Systems: Principles and Programming. — М.: «Вильямс», 2007. — 1152 с. — ISBN 978-5-8459-1156-8.
15. Нейлор К. Как построить свою экспертную систему / Chris Nailor. Build Your Own Expert System. John Wilei & Sons Ltd. Chichester. – М : Энергоатомиздат, 1991. – 286 с.
16. Морозов М.Н. Курс лекций по дисциплине "Системы искусственного интеллекта" [Електроний ресурс] / Морозов М.Н. // Лаборатория систем мультимедиа Марийский государственный технический университет — Режим доступу: http://khpi-iip.mipk.kharkiv.edu/library/ai/conspai/

ДОДАТОК

```
(deffunction ask-que (?q $?all-values)
  (printout t ?q)
  (bind ?ans (read))
  (if (lexemep ?ans)
    then (bind ?ans (lowcase ?ans)))
    (while (not (member ?ans ?all-values)) do
    (printout t ?que)
    (bind ?ans (read))
  (if (lexemep ?ans)
    then (bind ?ans (lowcase ?ans))))
?answer)
(deffunction y-or-n-q (?q)
  (bind ?res (ask-question ?q yes no y n))
  (if (or (eq ?res yes) (eq ? res y))
    then TRUE
  else FALSE))

(deffunction Pass-or-Issue-or-NA-test (?test)
  (bind ?res (ask-que ?test Pass Issue NA))
  (if ((eq ?res Pass))
    then (assert (?test Pass))
  )  (if ((eq ?res Issue))
    then (assert (?test Pass))
  ) (if ((eq ?res NA))
    then (assert (?test NA))
  )
(deffunction make-test (?test)
```

```
(bind ?tmp1 0)
(bind ?tmp1 2)
(bind ?fact_zagroz_ question_and_answer1 "Необхідні навички" "Не застосовується" "Немає технічних навичок" "Деякі технічні навички" "Просунутий користувач комп’ютера" "Просунутий користувач комп’ютера" "Навички проникнення безпеки")
(bind ?fact_zagroz_answer_score1 0 1 3 5 6)
(bind ?fact_zagroz_ question_and_answer2 "Мотив" "Не застосовується" "Низька або без винагороди" "Можлива винагорода" "Висока нагорода" )
(bind ?fact_zagroz_answer_score2 0 1 4 2)
(bind ?fact_zagroz_ question_and_answer3 " Можливість" "Повний доступ або необхідні дорогі ресурси" "Необхідний спеціальний доступ або ресурси" "Необхідний певний доступ або ресурси" "Не потрібен доступ чи ресурси" )
(bind ?fact_zagroz_answer_score3 0 4 7 9)
(bind ?fact_zagroz_ question_and_answer4 " Виконавці" "Не застосовується " "Системні адміністратори" "Користувачі внутрішньої мережі" "Автентифіковані користувачі"  "Анонімні користувачі Інтернету")
(bind ?fact_zagroz_answer_score4 0 2 4 5 6 9)
(bind ?fact_vrazlivosti_ question_and_answer1 " Знаходження" "Не застосовується
" " Практично неможливо " " Складно " "Легко"  "Доступні автоматизовані інструменти")
(bind ?fact_ vrazlivosti _answer_score1 0 1 3 5 9)
(bind ?fact_vrazlivosti_ question_and_answer2 "Реалізація " "Не застосовується " "Теоретично" " Складно " "Легко"  "Доступні автоматизовані інструменти")
(bind ?fact_ vrazlivosti _answer_score2 0 1 3 7 9)
(bind ?fact_vrazlivosti_question_and_answer3 "Інформованість" "Не застосовується
```

" " Невідомі знання " " Приховано знання " " Очевидні знання " " Загальнодоступні знання ")

(bind ?fact_ vrazlivosti _answer_score3 0 1 4 6 9)

(bind ?fact_vrazlivosti_ question_and_answer4 " Виявлення " "Активне виявлення в додатку" " Зареєстровано та переглянуто" "Увійшли без огляду" " Не зареєстровано" " Загальнодоступні знання ")

(bind ?fact_ vrazlivosti _answer_score4 0 1 3 8 9)

(bind ?fact_tehvplivu_ question_and_answer1 "Втрата кондефіційності" "Не застосовується

" " Розкриті мінімальні нечутливі дані " " Розкриті широкі нечутливі дані " " Розкриті широкі критичні дані " " Усі розкриті дані ")

(bind ?fact_ tehvplivu_answer_score1 0 2 6 7 9)

(bind ?fact_tehvplivu_question_and_answer2 " Втрата цілосності " "Не застосовується

" " Мінімально злегка пошкоджені дані " " Мінімально серйозно корумповані дані " " Обширні незначні корумповані дані " Значні з корумповані дані " Усі дані повністю пошкоджені")

(bind ?fact_ tehvplivu_answer_score2 0 1 3 5 7 9)

(bind ?fact_tehvplivu_question_and_answer3 " Втрата доступності " "Не застосовується

" " Мінімальні вторинні послуги перервано " " Мінімальні первинні послуги перервано " Перервано широкі первинні послуги " Усі послуги повністю втрачені ")

(bind ?fact_ tehvplivu _answer_score3 0 1 5 7 9)

(bind ?fact_tehvplivu_question_and_answer4 " Втрата звітності " "Не застосовується

" " Атака повністю простежується до особи " " Атака, можливо, простежується до особи " Атака абсолютно анонімна")

(bind ?fact_ tehvplivu _answer_score4 0 1 7 9)

(bind ?fact_vplivub_question_and_answer1 " Фінансовий " "Не застосовується" " Пошкодження шкоди менше, ніж для вирішення проблеми" " Незначний вплив на річний прибуток " " Значний вплив на річний прибуток " " Банкрутство ")

(bind ?fact_ vplivub _answer_score1 0 1 3 7 9)

(bind ?fact_vplivub_ question_and_answer2 " Репутаційний " "Не застосовується" " Мінімальний збиток " " Втрата основних рахунків " " Втрата доброї волі " Пошкодження торгової марки ")

(bind ?fact_ vplivub_answer_score2 0 1 4 6 9)

(bind ?fact_ vplivub_ question_and_answer3 " Невідповідність " "Не застосовується

" " Незначне порушення " " Явне порушення " Порушення високого профілю ")

(bind ?fact_ vplivub_answer_score3 0 2 5 7)

(bind ?fact_vplivub_question_and_answer4 " Політика кондефіційності" "Не застосовується" " Одна особа " " Сотні людей " " Тисячі людей " " Мільйони людей ")

(bind ?fact_ vplivub_answer_score4 0 3 5 7 9)

(bind ?res (ask-que (nth$ 1 ?fact_zagroz_ question_and_answer1) 1 2 3 4 5)

(bind tmp1 (+ ?tmp1 (nth$ (?res) ?fact_zagroz_answer_score1)))

(bind ?res (ask-que (nth$ 1 ?fact_zagroz_ question_and_answer2) 1 2 3 4 5)

(bind tmp1 (+ ?tmp1 (nth$ (?res) ?fact_zagroz_answer_score2)))

(bind ?res (ask-que (nth$ 1 ?fact_zagroz_ question_and_answer3) 1 2 3 4 5)

(bind tmp1 (+ ?tmp1 (nth$ (?res) ?fact_zagroz_answer_score3)))

(bind ?res (ask-que (nth$ 1 ?fact_zagroz_ question_and_answer4) 1 2 3 4 5)

(bind tmp1 (+ ?tmp1 (nth$ (?res) ?fact_zagroz_answer_score4)))

(bind ?res (ask-que (nth$ 1 ?fact_ vrazlivosti_ question_and_answer1) 1 2 3 4 5)

(bind tmp1 (+ ?tmp1 (nth$ (?res) ?fact_ vrazlivosti_answer_score1)))

(bind ?res (ask-que (nth$ 1 ?fact_ vrazlivosti_ question_and_answer2) 1 2 3 4 5)

```
  (bind tmp1  (+ ?tmp1 (nth$ (?res) ?fact_ vrazlivosti_answer_score2)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ vrazlivosti_ question_and_answer3) 1 2 3 4 5 )
  (bind tmp1  (+ ?tmp1 (nth$ (?res) ?fact_ vrazlivosti_answer_score3)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ vrazlivosti_ question_and_answer4) 1 2 3 4 5 )
  (bind tmp1  (+ ?tmp1 (nth$ (?res) ?fact_ vrazlivosti_answer_score4)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ tehvplivu_ question_and_answer1) 1 2 3 4 5 )
  (bind tmp2  (+ ?tmp2 (nth$ (?res) ?fact_ tehvplivu_answer_score1)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ tehvplivu_ question_and_answer2) 1 2 3 4 5 )
  (bind tmp2  (+ ?tmp2 (nth$ (?res) ?fact_ tehvplivu_answer_score2)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ tehvplivu_ question_and_answer3) 1 2 3 4 5 )
  (bind tmp2  (+ ?tmp2 (nth$ (?res) ?fact_ tehvplivu_answer_score3)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ tehvplivu_ question_and_answer4) 1 2 3 4 5 )
  (bind tmp2  (+ ?tmp2 (nth$ (?res) ?fact_ tehvplivu_answer_score4)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ vplivub_ question_and_answer1) 1 2 3 4 5 )
  (bind tmp2  (+ ?tmp2 (nth$ (?res) ?fact_ vplivub_answer_score1)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ vplivub_ question_and_answer2) 1 2 3 4 5 )
  (bind tmp2  (+ ?tmp2 (nth$ (?res) ?fact_ vplivub_answer_score2)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ vplivub_ question_and_answer3) 1 2 3 4 5 )
  (bind tmp2  (+ ?tmp2 (nth$ (?res) ?fact_ vplivub_answer_score3)))
  (bind ?res  (ask-que (nth$ 1 ?fact_ vplivub_ question_and_answer4) 1 2 3 4 5 )
  (bind tmp2  (+ ?tmp2 (nth$ (?res) ?fact_ vplivub_answer_score4)))
  (assert (?test likelihoodrisk (/ ?tmp1 8)))
    (assert (?test impactrisk (/ ?tmp2 8)))
  )

(defrule syst-banner ""
(declare (salience 10))
=>
(printout t crlf crlf)
```

```
(printout t "Експертна система пороведення аудиту))")
(printout t crlf crlf))

defrule find_score ""
  (?test likelihoodrisk ?score1)
  (?test impactrisk ?score2)
=>
  (if (and ( < ?score1 3) (< ?score2 3) )
   then
    (assert (?test not risk)))
  (if ((or (and ( < ?score1 3) (or (> ?score2 3) (< ?score2 6))) (and (or (> ?score1 3)
(< ?score1 6)) (< ?score2 3) )
   then
    (assert (?test low risk))
  (if ((or (and ( < ?score1 3) ( > ?score2 6)) (and( > ?score1 6) ( < ?score2 3))
   then
    (assert (?test moderate risk))
  (if ((or (and ( > ?score1 3) ( > ?score2 6)) (and( > ?score1 6) ( > ?score2 3))
   then
    (assert (?test hight risk)))
(defrule which_test_need ""
(declare (salience 9))
=>
  (if (y-or-n-q "Необхідно провести тестування збору інформації? (yes no)? ")
   then
    (assert (test info)))
  (if (y-or-n-q " Необхідно Тестування конфігурації та розгортання? (yes no)? ")
   then
    (assert (test conf)))
```

```
  (if (y-or-n-q " Необхідно Тестування перевірки даних? (yes no)? ")
   then
    (assert (test data)))
  (if (y-or-n-q " Необхідно Тестування криптографії сайту? (yes no)? ")
   then
    (assert (test crypto)))
  (if (y-or-n-q " Тестування на стороні клієнта? (yes no)? ")
   then
    (assert (test client)))
(assert (strart test)))
(defrule test1_1 ""
(test info)
(strart test)
=>
( Pass-or-Issue-or-NA-test ("Проведення пошуку та розвідки пошукової системи для пошуку витоку інформації  Використовуйте пошукову систему для пошуку мережевих діаграм та конфігурацій, облікових даних, вмісту повідомлення про помилку. Google Hacking, Sitedigger, Shodan, FOCA, Punkspider")))
(defrule test1_2 ""
(test info)
(strart test)
=>
( Pass-or-Issue-or-NA-test ("Відбиток веб-сервера. Знайдіть версію та тип запущеного веб-сервера, щоб визначити відомі вразливості та відповідні подвигиHttprint, Httprecon, Desenmascarame
")))

(defrule test1_3 ""
```

(test info)

(strart test)

=>

(Pass-or-Issue-or-NA-test ("Перегляньте метафайли веб-сервера на предмет витоку інформації Проаналізуйте robots.txt та визначте <META> теги з веб-сайту. Browser, curl, wget

")))

(defrule test1_3 ""

(test info)

(strart test)

=>

(Pass-or-Issue-or-NA-test ("Перегляньте коментарі та метадані на веб-сторінці щодо витоку інформації. Знайдіть конфіденційну інформацію з коментарів веб-сторінок та метаданих про вихідний код. Browser, curl, wget.

")))

(defrule test2_1 ""

(test conf)

(strart test)

=>

(Pass-or-Issue-or-NA-test ("Тестування обробки розширень файлів для чутливої інформації Пошук важливих файлів, інформацію (.asa, .inc, .sql, zip, tar, pdf, txt тощо). Nessus

")))

(defrule test2_2 ""

(test conf)

(strart test)

=>

(Pass-or-Issue-or-NA-test ("Файли резервного копіювання та невикористання для конфіденційної інформації Перевірте вихідний код JS, коментарі, файл кешу, файл резервної копії (.old, .bak, .inc, .src) та відгадайте ім'я файлу Nessus, Nikto, Wikto

")))

(defrule test2_3 ""

(test conf)

(strart test)

=>

(Pass-or-Issue-or-NA-test ("Перерахування інфраструктури та інтерфейсу адміністратора додатків Перерахуйте каталоги та файли, коментари та посилання у джерелі (/ admin, / administrator, / backoffice, / backend тощо), альтернативний порт сервера (Tomcat / 8080) Burp Proxy, dirb, Dirbuster, fuzzdb, Tilde Scanner

")))

(defrule test2_4 ""

(test conf)

(strart test)

=>

(Pass-or-Issue-or-NA-test ("Сувора безпека транспорту HTTP Визначте заголовок HSTS на веб-сервері за допомогою заголовка відповіді HTTP. завиток -s -D- https://domain.com/ | | grep Strict Burp Proxy, ZAP, curl

")))

(defrule test2_5 ""

(test conf)

(start test)

=>

(Pass-or-Issue-or-NA-test ("Перегляньте метафайли веб-сервера на предмет витоку інформації Проаналізуйте robots.txt та визначте <META> теги з веб-сайту. Browser, curl, wget
")))

(defrule test3_1 ""
(test data)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування XSS. Перевірте перевірку на вхід, замініть вектор використовуваний для ідентифікації XSS, на XSS з забрудненням параметра HTTP. Burp Proxy, ZAP, Xenotix XSS.
")))

(defrule test3_2 ""
(test data)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування на SQL-ін'єкцію. Union, Boolean, Error based, Out-of-band, Time delay. Burp Proxy (SQLipy), SQLMap, Pangolin, Seclists (FuzzDB)
")))

(defrule test3_3 ""
(test data)
(start test)
=>

(Pass-or-Issue-or-NA-test ("Тестування Oracle. Визначте URL-адреси для веб-застосунків PL / SQL, Доступ із пакетами PL / SQL, Обхід списку виключень PL / SQL, Введення SQL Orascan, SQLInjector
")))

(defrule test3_4 ""
(test data)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування MySQL. Визначте версію MySQL, Єдину цитату, Інформаційну схему, Файл читання / запису. SQLMap, Mysqloit, Power Injector
")))
(defrule test3_5 ""
(test data)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування доступу MS Перерахуйте стовпчик на error-based (Group by), Отримати схему бази даних поєднати з fuzzdb. SQLMap
")))

(defrule test3_6 ""
(test data)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування на ін'єкцію NoSQL Визначте бази даних NoSQL, передайте спеціальні символи ('"\; {}), Атака із зарезервованою назвою змінної, оператором. NoSQLMap

")))

(defrule test3_7 ""

(test data)

(start test)

=>

(Pass-or-Issue-or-NA-test ("Тестування на ін'єкцію XML Перевірте XML мета символи', "", <>, <! - / ->, &, <! [CDATA [/]]>, XXE, TAG Burp Proxy, ZAP, Wfuzz

")))

(defrule test4_1 ""

(test crypto)

(start test)

=>

(Pass-or-Issue-or-NA-test ("Тестування на Padding Oracle. Перевірте данні трьох станів: Шифровий текст розшифровується, а отримані дані є правильними. Шифровий текст розшифровується, отримані дані є неправильними але викликаються деякий обробник виняткових подій або механізм роботи з помилками в логіці програми. Розшифровка тексту шифру не вдається через помилки заміщення. " PadBuster, Poracle, python-paddingoracle, POET

")))

(defrule test4_2 ""

(test crypto)

(start test)

=>

(Pass-or-Issue-or-NA-test ("Тестування на чутливу інформацію, що надсилається через незашифровані канали Перевірте конфіденційні дані під час передачі: Інформація, що використовується для автентифікації (наприклад, Уповноважені дані, PIN-коди, Сесія ідентифікатори, маркери,

файли cookie ...) Інформація, захищена законами, нормативними актами або конкретними організаційними
політика (наприклад, кредитні картки, дані клієнтів) Burp Proxy, ZAP, Curl
")))

(defrule test5_1 ""
(test crypto)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування сценарію крос-сайтів на основі DOM. Тест на введення даних користувачів, отриманих від клієнтських об’єктів JavaScriptBurp Proxy, ZAP
")))
(defrule test5_2 ""
(test crypto)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування на ін'єкцію HTML. Надіслати шкідливий HTML-код: ? user = <img% 20src = 'aaa'% 20onerror = попередження (1)>Burp Proxy, ZAP
")))
(defrule test5_2 ""
(test crypto)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування на перенаправлення на сторону клієнта. Змінити ненадійне введення URL-адреси на шкідливий сайт: (Відкрити перенаправлення)? redirect = www.fake-target.site.Burp Proxy, ZAP
")))

(defrule test5_3 ""
(test crypto)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування на Clickjacking. Виявіть, чи веб-сайт вразливий, завантажившись у рамку iframe, створіть просту веб-сторінку, що включає рамку, що містить ціль. Burp Proxy, ClickjackingTool
")))
(defrule test5_4 ""
(test crypto)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування на Clickjacking. Виявіть, чи веб-сайт вразливий, завантажившись у рамку iframe, створіть просту веб-сторінку, що включає рамку, що містить ціль. Burp Proxy, ClickjackingTool
")))

(defrule test5_5 ""
(test crypto)
(start test)
=>
(Pass-or-Issue-or-NA-test ("Тестування WebSockets. Визначте, що програма використовує WebSockets, перевіривши схему ws: // або wss: // URI. Перевірте походження, конфіденційність та цілісність, автентифікацію, авторизацію, санітарію введення Burp Proxy, Chrome, ZAP, WebSocket Client
")))