Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Створення програмного забезпечення для аналізу та безпеки вебсерверів

Виконав: студент II курсу, групи СБм-61
спеціальності 125 Кібербезпека
(шифр і назва спеціальності)

(підпис) Назарук О.Ю. (прізвище та ініціали)

Керівник (підпис) Карпінський М.П. (прізвище та ініціали)

Нормоконтроль (підпис) Лечаченко Т.А. (прізвище та ініціали)

Завідувач кафедри (підпис) Загородна Н.В. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Назаруку Олександру Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення програмного забезпечення для аналізу та безпеки вебсерверів

Керівник роботи Карпінський Микола Петрович, д.т.н., проф. кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_16_» _листопада_ 2023 року № 4/7-1061
2. Термін подання студентом завершеної роботи 14 грудня 2023р.
3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити):
   1. Протокол передачі даних - http
   2. Основні загрози в мережі інтернет
   3. Створення програми для підвищення рівня безпеки вебсерверів "ВАТ"
   4. Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 титульна сторінка. 2 мета. завдання дослідження. наукова новизна
3 Запит-відповідь протоколу http 4 Запит-відповідь протоколу http через
proxy-сервер 5 Загальна класифікація загроз АБС 6 Класифікація кібератак
7 Методи виявлення атак 8 Оцінка інформативності параметрів
9 Методи та методики оцінки ризиків 10 Методика CRAMM
11 Методика FAIR 12 Вхідні дані-база KDD99 13 База даних NSL-KDD
14 Створення програми для підвищення рівня безпеки вебсерверів "ВАТ"

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 16 листопада 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	16.11.2023-17.11.2023	*Виконано*
2.	Підбір наукових джерел про загрози для вебсерверів	18.11.2023-20.11.2023	*Виконано*
3.	Переклад та опрацювання наукових джерел про		
	дослідження вразливостей вебсерверів та методів		
	захисту	21.11.2023-23.11.2023	*Виконано*
4.	Розробка програмного забезпечення для		
	підвищення рівня безпеки вебсерверів організацій		
	на основі проведеного дослідження	24.11.2023-27.11.2023	*Виконано*
5.	Оформлення розділу «Протокол передачі даних		
	http»	28.11.2023-30.11.2023	*Виконано*
6.	Оформлення розділу «Основні загрози в мережі		
	Інтернет»	01.12.2023-04.12.2023	*Виконано*
7.	Оформлення розділу «Створення програми для		
	підвищення рівня безпеки вебсерверів ВАТ»	05.12.2023-07.12.2023	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	08.12.2023-09.12.2023	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в надзви-		
	чайних ситуаціях»	10.12.2023-11.12.2023	*Виконано*
10.	Оформлення кваліфікаційної роботи	12.12.2023-13.12.2023	*Виконано*
11.	Нормоконтроль	14.12.2023-15.12.2023	*Виконано*
12.	Перевірка на плагіат	09.12.2023	*Виконано*
13.	Попередній захист кваліфікаційної роботи	16.12.2023	*Виконано*
14.	Захист кваліфікаційної роботи	26.12.2023	

Студент _____ Назарук О.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Карпінський М.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення програмного забезпечення для аналізу та безпеки вебсерверів // Кваліфікаційна роботи рівня «Магістр» // Назарук Олександр Юрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра кібербезпеки, група СБм–61 // Тернопіль, 2023 // с. - , рис. - , бібліогр. – .

Ключові слова: ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАММУВАННЯ, ВЕБ-БЕЗПЕКА, КІБЕРАТАКА.

Об'єктом кваліфікаційної роботи є процес аналізу аналізу рівня безпеки вебсерверів.

Предмет кваліфікаційної роботи – створення програмного забезпечення для аналізу та безпеки вебсерверів.

Мета кваліфікаційної роботи – створення легкого у використовуванні сучасного і повнофункціонального програмного продукту для аналізу рівня безпеки вебсерверів.

Створене програмне забезпечення дозволяє підвищити рівень безпеки захисту інформації на серверах. Програмний продукт також має задані властивості котрі дають змогу в декілька секунд , отримати повну інформацію про статус роботи серверів.

ANNOTATION

Software Development for Web Servers Analysis and Safety // Qualification paper of the educational level “Master” // Oleksandr Nazaruk // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cyber Security, SBm-61 group // Ternopil, 2023 // P. , fig. - , tables - , annexes - , references - .

Key words: OBJECT-ORIENTED DESIGN, CYBERSECURITY OBJECT-ORIENT-VANE PROGRAMMING, WEB SECURITY, CYBERATTACK.

The object of the qualification work is the process of analyzing the security level of web servers.

The subject of the qualification work is the creation of software for the analysis and security of web servers.

The goal of the qualification work is to create an easy-to-use modern and fully functional software product for analyzing the security level of web servers.

The created software allows to increase the security level of information protection on servers. The software product also has specified properties that allow you to get complete information about the status of the servers in a few seconds.

ЗМІСТ

ВСТУП

Метою даної кваліфікаційної роботи є створення програмного забезпечення, яке дає змогу підвищити рівень безпеки серверів.

Об’єктом кваліфікаційної роботи є запобігання вірусних атак.

Предметом кваліфікаційної роботи є програма, що знаходить зайві або зовсім непротрібні заголовки-відповіді від вебсерверу.

Головними задачами кваліфікаційної роботи є реалізація представлення статусу серверів і їх безпеки в інтернет просторі, Знаходження слабких місць які в свою чергу можуть бути використовані кіберзлочинцями.

Робота основана на ідеї створення застосунку, що дає змогу завдяки відправці http-запитів на сторону серверу, отримати повну інформацію котра надається з відповіді від серверу. Користувач має можливість переглядати значення головних заголовків котрі будуть представлені програмою “BAT” у комфортному для людини вигляді а також побачити на що треба звернути увагу для запобігання кібератак .

В основі функціонування програми лежить алгоритм зчитування інформації із онлайн-документації RFC лежить, на підставі чого програма “BAT” може встановити аналіз статусу безпеки серверів. Саме отримання інформації із документації, дає змогу переконатись у коректності отриманих даних через програму “BAT”.

У першій частині пояснювальної записки до кваліфікаційної роботи описуються принцип роботи і структура HTTP протоколу.

У другій частині кваліфікаційної роботи розглянуті основні загрози в мережі інтернет.

Третя частина присвячена створенню програми для підвищення рівня безпеки вебсерверів.

1 ПРОТОКОЛ ПЕРЕДАЧІ ДАНИХ – HTTP

1.1 Характеристика протоколу HTTP

HTTP - це протокол передачі гіпертексту на прикладному рівні, який використовується для обміну інформацією в розподілених, спільних і багатофункціональних інформаційних системах. Початок використання HTTP в World Wide Web (WWW) відзначається з 1990 року. Першою версією HTTP був простий протокол для передачі необроблених даних через Інтернет. Однак швидке збільшення кількості додатків, повністю сумісних з HTTP/1.0, призвело до необхідності введення нової версії протоколу, яка мала додаткові функціональність, спрямовану на уніфікацію цих додатків у єдиний стандарт.

Протокол HTTP/1.1 містить більш суворі вимоги, ніж HTTP / 1.0, що гарантують більш надійну роботу.

Великі інформаційні системи вимагають більшої кількості функціональних можливостей, ніж просто завантаження інформації, включаючи пошук і модифікацію даних за допомогою зовнішніх інтерфейсів.

HTTP надає відкритий (open-ended) набір методів, які засновані на системі посилань, які забезпечуються URI (Універсальними Ідентифікаторами Ресурсів). URI можуть ідентифікувати як розташування (URL), так і ім'я (URN) ресурсу, до якого застосовується даний метод.

Повідомлення передаються у форматі, що нагадує формат, який використовується в електронній пошті, відповідно до MIME (Багатоцільових Розширень Електронної Пошти). Крім того, HTTP служить як загальний протокол зв'язку між агентами користувачів та проксі-серверами / шлюзами або іншими Інтернет-сервісами, такими як SMTP, NNTP, FTP, Gopher і WAIS. Отже, HTTP встановлює основи для доступу до ресурсів у різних середовищах для різних застосунків.

1.2 Загальний опис

HTTP протокол - це система запитів та відповідей. Коли клієнт відправляє запит серверу через з'єднання, він включає метод запиту, URL, версію протоколу, а також MIME-подібне повідомлення з модифікаторами запиту, інформацією про клієнта і, іноді, тілом запиту. Відповідь сервера містить рядок стану з версією протоколу, кодом статусу (успіх чи помилка) і MIME-подібним повідомленням, що включає деталі про сервер, метаінформацію об'єкта та, можливо, тіло об'єкта.

В більшості випадків, агент користувача ініціює HTTP з'єднання, які складаються з одного запиту до ресурсу на основному сервері. Найпростіший випадок вимагає лише одного з'єднання між агентом користувача і основним сервером (рис.1.1).

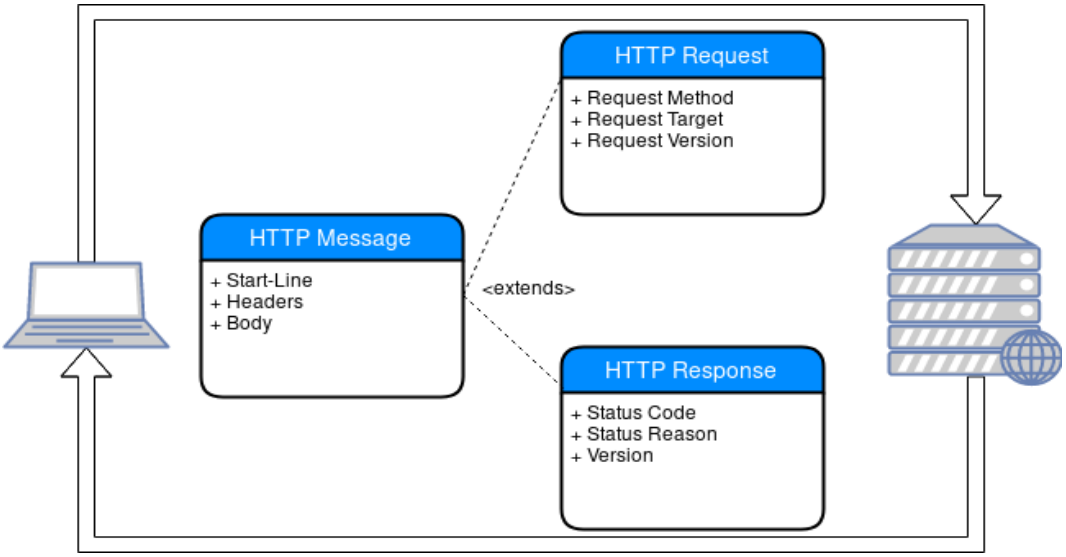

Рисунок 1.1 – Запит-відповідь протоколу HTTP

У більш складних випадках, коли в процесі взаємодії запитів та відповідей задіяні один або кілька інтермедіарів, виникають додаткові складнощі. Ці інтермедіари поділяються на три основні категорії: проксі-сервери, шлюзи і тунелі. Проксі-сервер функціонує як посередник, приймаючи запити на певний URI в абсолютній формі, модифікуючи ці запити частково або повністю, і перенаправляючи їх до сервера, вказаного в URI.

Шлюз виступає як інтермедіар на вищому рівні відносно іншого сервера або серверів, перекладаючи запити в протокол цього основного сервера при необхідності. Тунель, у свою чергу, діє як реле між двома з'єднаннями, передаючи повідомлення без змін. Тунелі часто використовуються для забезпечення з'єднання через інтермедіар, наприклад, брандмауер, який не може інтерпретувати зміст повідомлень.

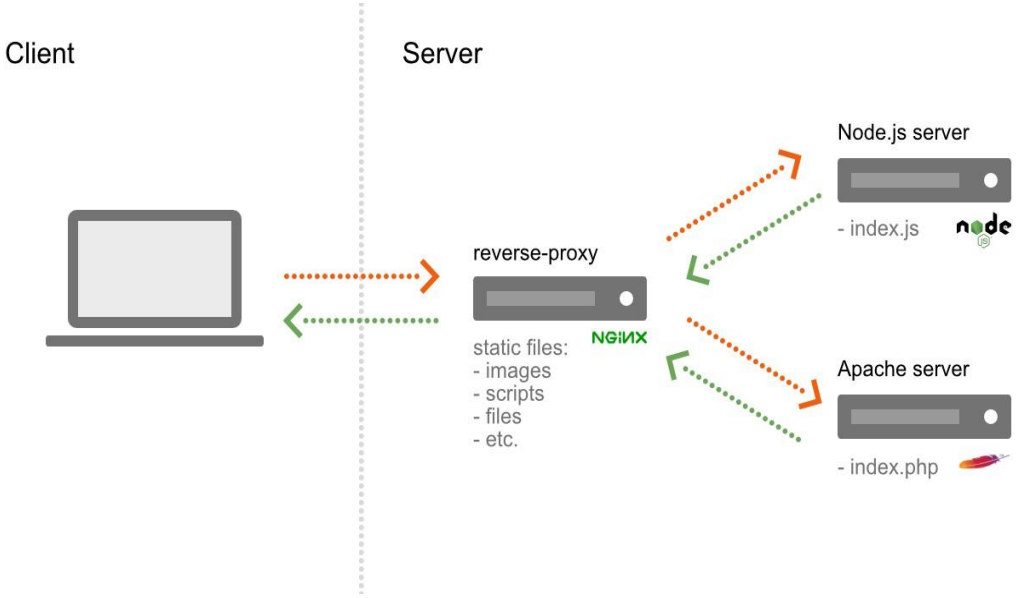

Рисунок 1.2 - Запит-відповідь протоколу HTTP через proxy-сервер

На рис. 1.2 Реверс-проксі (reverse-proxy) є проміжним засобом між агентом користувача та початковим сервером, через який запити та відповіді передаються через окремі з'єднання. Ця особливість має важливе значення, оскільки деякі параметри HTTP застосовні тільки до з'єднання з найближчим не-тунельним вузлом, деякі - лише до кінцевих точок ланцюжка, а деякі - до всіх з'єднань у ланцюжку.

Зазвичай HTTP-з'єднання встановлюються за допомогою протоколу TCP/IP, зі стандартним портом TCP 80, хоча інші порти, такі як 8080 чи 8081, також можуть використовуватися. HTTP вимагає лише надійної передачі даних, тому може використовувати будь-який протокол, що гарантує надійність передачі даних. Деталі структури запиту та відповіді HTTP/1.1 обробляються на

рівні транспортного модуля даних, що використовується у вказаному протоколі, і це питання не визначається самим HTTP-протоколом.

У більшості реалізацій HTTP/1.0 використовувалось нове з'єднання для кожного обміну запитом/відповіддю. У HTTP/1.1, встановлене з'єднання може використовуватися для одного або декількох обмінів запитом/відповіддю, хоча з'єднання може бути закрито з різних причин.

1.2.1 Версія HTTP

HTTP використовує номерування версій у форматі "<major>.<minor>" для вказівки версії протоколу. Ця стратегія дозволяє відправнику вказати формат повідомлення і своє розуміння для подальшого зв'язку через HTTP перед отриманням відповіді. Якщо додаються компоненти повідомлення, які не впливають на процес зв'язку, або компоненти, які додаються лише для розширення значення поля, номер версії не змінюється.

Номер <minor> зростає, коли внесені зміни до протоколу додають можливості, які не змінюють загальний алгоритм аналізу повідомлень, але розширюють семантику повідомлення. В такому випадку стають можливими додаткові функції відправника. Якщо змінюється формат повідомлення протоколу, збільшується номер <major>.

Версія HTTP повідомлення вказується у полі HTTP-version у першому рядку повідомлення, і має такий формат: "HTTP" "/" 1*DIGIT "." 1*DIGIT.

Major і minor номери повинні оброблятися як окремі цілі числа, і кожен з них може складатися з більш ніж однієї цифри. Таким чином, HTTP/2.4 є більш низькою версією, ніж HTTP/2.13, яка, в свою чергу, нижча за HTTP/12.3. Нулі в номерах можуть бути проігноровані одержувачем і не враховуватися.

Додатки, які надсилають запити або відповіді, які відповідають специфікації HTTP/1.1, повинні вказувати версію HTTP як "HTTP/1.1". Використання цієї версії вказує, що додаток є принаймні умовно сумісним з цією специфікацією.

HTTP версія програми визначається як найвища версія HTTP, з якою додаток є принаймні умовно сумісним.

Додатки, що реалізують проксі-сервера і шлюзи, повинні обробляти протокольні повідомлення різних версій. Починаючи з моменту, коли версія протоколу вказує можливості відправника, проксі-сервер / шлюз ніколи не повинен посилати повідомлення, версія яких більше, ніж HTTP версія відправника; якщо отримана більш висока версія запиту, то проксі-сервер / шлюз повинен або знизити версію запиту, повернувши повідомлення про помилку, або переключитися на тунельний поведінку. У запитів, версія яких нижче, ніж HTTP версія проксі-сервера / шлюзу можна перед пересиланням збільшити версію; відповідь проксі-сервера / шлюзу на цей запит повинен мати ту ж саму major версію, що і запит.

1.2.2 Загальний синтаксис HTTP

Схема "HTTP" використовується для отримання доступу до мережевих ресурсів через протокол HTTP. Ця частина визначає конкретний синтаксис і значення для URL-адрес, які використовують протокол HTTP.

```
http_URL = "http:" "//" host [ ":" port ] [ abs_path ]
host = <допустиме доменне ім’я машини чи IP адрес (в
десятковій формі), як визначено в розділі 2.1 RFC 1123>
port = *DIGIT
```

1.2.3 URI

URI відомі під різними назвами, такими як веб-адреси, Універсальні Ідентифікатори Документів, Універсальні Ідентифікатори Ресурсів (URI). Крім того, URI може бути комбінацією Uniform Resource Locators (URL) і Uniform Resource Names (URN). Протокол HTTP визначає URL як рядок певного формату, який ідентифікує ресурс за допомогою імені, місця розташування або інших характеристик.

1.2.4 Загальний синтаксис URI.

URI в HTTP можуть бути використані в двох формах, в залежності від контексту: абсолютній формі (absolute URI) і відносній формі (relative URI). Ці дві варіанти відрізняються тим, що абсолютні URI завжди мають починатися з імені схеми, вказаного з двокрапкою (див лістинг 1.1).

Лістинг 1.1 – Деталі налаштування URI

```
URI = ( absoluteURI | relativeURI ) [ "#" fragment ]
absoluteURI = scheme ":" *( uchar | reserved )
relativeURI = net_path | abs_path | rel_path
net_path = "//" net_loc [ abs_path ] abs_path = "/"
rel_path rel_path = [ path ] [ ";" params ] [ "?" query ]
path = fsegment *( "/" segment ) fsegment = 1*pchar segment
= *pchar
params = param *( ";" param ) param = *( pchar | "/" )
scheme = 1*( ALPHA | DIGIT | "+" | "-" | "." ) net_loc = *(
pchar | ";" | "?" )
query = *( uchar | reserved ) fragment = *( uchar |
reserved )
pchar = uchar | ":" | "@" | "&" | "=" | "+" uchar =
unreserved | escape unreserved = ALPHA | DIGIT | safe |
extra | national
escape = "%" HEX HEX reserved = ";" | "/" | "?" | ":" | "@"
| "&" | "=" | "+" extra = "!" | "*" | "'" | "(" | ")" | ","
safe = "$" | "-" | "_" | "." unsafe = CTL | SP | <"> | "#"
| "%" | "<" | ">" national = <любой ОСТЕТ за виключенням
ALPHA, DIGIT, reserved, extra, safe, и unsafe октетів>
```

Деталі URL описані у RFC 1738 та 1808. Вони можуть включати символи, які офіційно не дозволені, оскільки деякі HTTP сервери приймають URI з незабороненими символами, що може призвести до помилок у HTTP проксі. Довжина URI для HTTP серверів не обмежена, вони повинні обробляти URI

будь-якої довжини, хоча сервери можуть повернути помилку 414, якщо URI надто довгий. Слід бути уважними з URI понад 255 байтів, адже старі клієнти можуть їх некоректно обробляти.

1.2.5 Кодові таблиці (character sets).

HTTP використовує термін “кодова таблиця”, тобто використовує одну або кілька таблиць для перетворення послідовності октетів в послідовність символів. Варто відзначити, що однозначне перетворення в зворотному напрямку не потрібно, і що не всі символи можуть бути доступні в даній кодової таблиці, і що кодова таблиця може забезпечувати більш ніж одну послідовність октетів для подання специфічних символів.

Це визначення допускає різні види кодування символів, від простих однотаблічних відображень типу US-ASCII до складних методів, перемикаючих таблиці, на зразок тих, які використовують методики ISO 2022. Однак визначення, пов'язане з ім'ям кодової таблиці MIME повинно повністю визначати відображення, яке перетворює октети в символи. Зокрема використання зовнішньої інформації профілювання для визначення точного відображення забороняється.

Кодові таблиці HTTP ідентифікуються лексемами, що не чутливими до регістру. Повний набір лексем визначено реєстром кодових таблиць IANA

```
charset = token
```

Хоча HTTP дозволяє використовувати в якості значення charset довільну лексему, будь-яка лексема, яка має визначене значення в реєстрі кодових таблиць IANA, повинна представляти набір символів, визначений в даному реєстрі. Додатків слід обмежити використання символьних наборів тими, які визначені в реєстрі IANA.

1.2.6 Кодування вмісту (content codings).

Кодування вмісту використовується перш за все для стиснення або іншого корисного перетворення документа без втрати ідентифікації основного медіатіпа та інформації. Часто, об’єкт зберігається в кодованому вигляді, потім передається, а потім декодується одержувачем.

```
content-coding = token
```

Всі значення кодування вмісту (content-coding) не чутливі до регістру. HTTP / 1.1 використовує значення кодування вмісту (content-coding) в полях заголовка Accept-Encoding і Content-Encoding. Хоча значення описує кодування вмісту, але, що більш важливо – воно вказує, який механізм декодування потрібно для зворотного процесу.

Internet Assigned Numbers Authority (IANA) діє як реєстр для значень лексем кодування вмісту (content-coding). Спочатку реєстр містив такі лексеми:

– **gzip** Формат кодування, що виробляє стиснення файлу програмою "gzip" (GNU zip), описаний в RFC 1952. Це формат Lempel-Ziv кодування (LZ77) з 32 розрядним CRC.

– **compress** Формат кодування, вироблений спільною програмою "compress" для стиснення UNIX файлів. Це формат адаптивного Lempel-Ziv-Welch кодування (LZW).

Звичайно, використовувати назви програм для ідентифікації форматів кодування не бажано і може перетинатися з форматами, які виникнуть після. Їх використання пояснюється історичною практикою. Для сумісності з попередніми реалізаціями HTTP, додатки повинні розглядати "x-gzip" і "x-compress" як еквіваленти "gzip" і "compress" відповідно.

– **deflate** Формат zlib, певний в 1950, в комбінації з механізмом стиснення "deflate", описаним в RFC 1951.

Нова лексема значення кодування вмісту (content-coding) повинна бути зареєстрована; щоб забезпечити взаємодію між клієнтами і серверами,

специфікація алгоритму кодування вмісту, необхідного для визначення нового значення, повинна бути відкрито опублікована і адекватна для незалежної реалізації, а також відповідати меті кодування вмісту певного в цьому розділі.

1.3 HTTP повідомлення (HTTP Message)

HTTP повідомлення діляться на запити клієнта серверу і відповіді сервера клієнтові.

```
HTTP-message = Request | Response ; сообщения HTTP/1.1
```

Повідомлення запиту і відповіді користуються стандартизованим форматом повідомлень RFC 822 для передачі об'єктів, які містять корисну інформацію (так зване повідомлення). Обидва види повідомлень мають таку ж структуру: спочатку іде початковий рядок (start-line), за ним слідує одне або кілька полів заголовка (іноді просто називають "заголовками"), після чого іде порожній рядок (це рядок, який складається з символів CRLF), який позначає кінець заголовкової частини, і, можливо, саме тіло повідомлення.

```
generic-message = start-line *message-header CRLF [
message-body ]
start-line = Request-Line | Status-Line
```

В інтересах спроможності до помилки, серверам слід ігнорувати всі порожні рядки, отримані перед рядком запиту (Request-Line). Іншими словами, якщо сервер читає потік протоколу і на самому початку повідомлення отримує CRLF, то йому слід цей CRLF ігнорувати.

Деякі помилкові реалізації HTTP / 1.0 клієнтів генерують додаткові CRLF після запиту POST. Варто знову повторити, що це явно заборонено нормальної записом Бекуса-Наура. HTTP / 1.1 клієнт не повинен додавати додаткові CRLF перед запитом і після нього.

1.3.1 Заголовки повідомлень

Заголовки HTTP дозволяють клієнту та серверу передавати додаткову інформацію у запиті чи відповіді. Заголовок запиту складається з нечутливих до регістру імен, за якими йде двокрапка ':', а потім їх значення (без розбиття на рядки). Пробіли перед значеннями ігноруються.

Користувальницькі значення заголовків можуть бути додані за допомогою префіксу 'X-' . Однак ця домовленість більше не підтримується з червня 2012, через незручність, яку вона викликала, коли нестандартизовані поля стали стандартними у RFC 6648; інші значення перераховані у IANA registry, оригінальний зміст якого був виначений у RFC 4229. IANA також підтримує реєстр запропонованих нових значень заголовків повідомлення HTTP.

Заголовки можуть бути згруповані, відповідно до їхнього змісту:

1. General header: Заголовки, що застосовуються як для запиту, так і відповіді, але не мають ніякого відношення до даних, що передаються у тілі.
2. Request header: Заголовки, що містять більше інформації про ресурс, що запитується, чи самого клієнта.
3. Response header: Заголовки з додатковою інформацією про відповідь, наприклад, її розташування або сам сервер (ім'я, версія тощо).
4. Entity header: Заголовки, що містять больше інформації про тіло сутності, наприклад, довжину її контенту або MIME-тип.

Заголовки також можуть бути згруповані за тим, як проксі обробляє їх:

End-to-end (точка-точка або наскрізні) заголовки

Ці заголовки повинні бути передані кінцевому отримувачу повідомлення; серверу, що оброблює запит, або клієнту, який отримує відповідь. Проміжний проксі має повторно передати наскрізні заголовки незмінними, а кеш повинен зберегти їх.

Hop-by-hop (крок-за-кроком або проміжні) заголовки

Ці заголовки мають значення лише для одного з’єднання транспортного рівня та не повинні бути повторно передані через проксі або кеш. Зазначте, що

лише проміжні заголовки можуть бути встановлені за допомогою загального заголовку Connection.

Наступний лист підсумовує заголовки HTTP за категорією їх використання:

Забезпечення автентифікації:

1. WWW-Authenticate

Визначають методи автентифікації, що мають бути використані для отримання доступу до ресурсу.

2. Authorization

Містить облікові данні для автентифікації агента користувача сервером.

3. Proxy-Authenticate

Визначає метод автентифікації, який має бути використаний для отримання доступу до ресурсу через проксі сервер.

4. Proxy-Authorization

Містить облікові данні для автентифікації агента користувача проксі сервером.

Забезпечення доступності:

1. Access-Control-Allow-Origin

Вказує, чи можна надати спільний доступ до відповіді.

2. Access-Control-Allow-Credentials

Вказує, чи можна надати відповідь на запит, коли позначка облікових даних істинна.

3. Access-Control-Allow-Headers

Використовується для відповіді на запит перед друком, щоб вказати, які заголовки HTTP можна використовувати під час створення конкретного запиту.

4. Access-Control-Allow-Methods

Визначає метод або методи, дозволені під час доступу до ресурсу у відповідь на запит перед друком.

5. Access-Control-Expose-Headers

Вказує, які заголовки можна викрити як частину відповіді, перерахувавши їхні імена.

6. Access-Control-Max-Age

Вказує, як довго можна кешувати результати запиту перевірки перед друком.

7. Access-Control-Request-Headers

Використовується під час видачі запиту перед друком, щоб сервер знав, які заголовки HTTP буде використовуватися під час фактичного запиту.

8. Access-Control-Request-Method

Використовується під час видачі запиту перед друком, щоб сервер знав, який метод HTTP буде використовуватися під час фактичного запиту.

9. Origin

Вказує, звідки бере початок отримання.

Забезпечення відстеження:

1. DNT

Використовується для вираження уподобань відстеження користувача.

2. TK

Указує на стан відстеження, застосований до відповідного запиту.

Забезпечення цілісності:

1. Політика безпеки вмісту (CSP)

Керує ресурсами, які агенту користувача дозволено завантажувати для вказаної сторінки.

2. Content-Security-Policy-Report-Only

Дозволяє веб-розробникам експериментувати з політикою, відстежуючи (але не застосдуючи) їх ефекти. Ці звіти про порушення складаються з документів JSON, надісланих через запит HTTP до вказаного URI.POST

3. Public-Key-Pins (HPKP))

Пов’язує певний криптографічний відкритий ключ із певним веб-сервером для зменшення ризику атак MITM з підробленими сертифікатами.

4. Public-Key-Pins-Report-Only

Надсилає звіти звіт-uri, указаний у заголовку і все ще дозволяє клієнтам підключитися до сервера, навіть якщо закріплення порушується.

5. Strict-Transport-Security(HSTS)

Примусовий зв’язок за допомогою HTTPS замість HTTP.

6. Upgrade-Insecure-Requests

Надсилає на сервер сигнал, який вказує на бажання клієнта на зашифровану і аутентифіковану відповідь, і що він може успішно обробити директиву upgrade-insecure-requests

7. X-Content-Type-Options

Вимикає “сніфер” MIME і змушує браузер використовувати тип, вказаний у типі вмісту.

8. Параметри рамки X (XFO)

Указує, чи слід дозволити переглядачу відображати сторінку у <frame>, <iframe> або <object>

9. X-XSS-Protection

Вмикає фільтрацію розміщення XSS-сценаріїв.

1.3.2 Тіло HTTP-повідомлення.

Якщо присутнє тіло повідомлення (message-body), то воно передає дані об'єкта і може відрізнятись від тіла об'єкта при кодуванні Transfer-Encoding.

message-body = entity-body | <entity-body закадованого відповідно Transfer-Encoding>

Поле Transfer-Encoding використовується для вказівки, як будь-яке кодування передачі впливає на безпеку і правильність передачі повідомлень. Це поле є властивістю повідомлення, а не об'єктом, і, отже, його можуть додавати або видаляти будь-які додатки у послідовності записів або відповідей.

Правила, які визначають, чи може бути тіло повідомлення включене в запиті або відповіді, різняться для запитів і відповідей.

Щоб дізнатись, чи присутнє тіло повідомлення в запиті відбувається через додавання поля заголовка Content-Length або Transfer-Encoding до заголовків

запиту. Тіло повідомлення може бути додано в запит лише тоді, коли метод запиту передбачає наявність тіла об'єкта.

Питання про те, чи тіло повідомлення має бути включене у відповідь, залежить від методу запиту і коду стану відповіді. Усі відповіді на запит з методом HEAD не повинні містити тіло повідомлення, навіть якщо поля заголовка об'єкта вказують на його наявність. Відповіді з інформаційними кодами стану 1xx, кодом 204 (Ні вмісту, No Content) і кодом 304 (Не модифікований, Not Modified) також не повинні включати тіло повідомлення. Всі інші відповіді мають містити тіло повідомлення, навіть якщо воно має нульову довжину.

Для проведення повнофункціонального якісного тестування програмного продукту з боку працездатності , був розроблений окремий файл run.py , який надає змогу перевірити , чи були встановлені всі необхідні залежності на систему користувача .

Після тестування додатку та аналізу списку протестованих функцій і виявлених дефектів, ми можемо зробити висновок щодо якості додатку. Наш підхід включає тестування навантаження, властивостей, інсталяцію, регресію та графічний інтерфейс користувача.

Результати тестування програмного забезпечення наведено у табл. 1.1.

Таблиця 1.1 – Перевірка роботи адмінської частини веб-додатку

№ п/п	Виконання тесту	Очікуваний результат	Отриманий результат	Прохо-дження тесту
1.	Після запуску програми, отримати довідку щодо шаблону виконання команд	Отримання підказки котра дає можливість швидко зрозуміти як працювати з даним інструментом	Отримаємо результат с ключами для роботи с командами користувача	Тест пройдений
2.	Отримати вихідні дані в json-форматі	Отримання масиву даних в JSON	Отриманий масив	Тест пройдений

1.3.3 Довжина тіла HTTP-повідомлення.

Якщо тіло повідомлення (message-body) присутнє в повідомленні, його довжина визначається за одним із наступних методів (в порядку пріоритету):

Будь-яка відповідь, яку мав би включати тіло повідомлення (message-body) (наприклад відповіді з кодами стану 1xx, 204, 304 і всі відповіді на запит HEAD) завжди завершується символом нового рядка після полів заголовка, незалежно від полів заголовка об’єкта (entity-header fields), представлених в повідомленні.

1. Якщо поле заголовка Transfer-Encoding присутній і вказує на застосування кодування передачі “chunked”, то довжина визначається кодуванням по шматках (chunked encoding).

2. Якщо поле заголовка Content-Length присутній, то його значення представляє довжину тіла повідомлення (message-body) в байтах.

3. Якщо повідомлення використовує медіатіп “multipart / byteranges”, який саморазгранічен, то він і визначає довжину. Цей медіа тип не повинен використовуватися, якщо відправник не знає, чи здатний одержувач його обробити; присутність в запиті заголовка Range з декількома специфікаторами діапазонів байтів (byte-range) має на увазі, що клієнт може аналізувати multipart / byteranges відповіді.

4. Довжина визначається закриттям з’єднання сервером.

Для сумісності з HTTP/1.0 Додатками HTTP/1.1 запиті, що містять Тіло повідомлення (message-body) повинні включать допустиме поле заголовка Content-Length, поки НЕ відомо, что сервер є HTTP / 1.1 сумісним. якщо запит містіть тіло повідомлення (message-body), и Content-Length не вказано, серверу слід послати відповідь з кодом стану 400 (Зіпсованій запит, Bad Request), Якщо він не может візначіті довжина повідомлення, або з якихось кодом стану 411 (Потрібно довжина , Length Required), Якщо він наполягає на отріманні Content-Length.

Всі HTTP/1.1 додатка, Які отримуються об’єкти, повінні розуміті кодування передачі типу "chunked", таким чином дозволяється использование

даного механізму для таких Повідомлень, довжина якіх не може бути отримана заздалегідь.

1.6 Запит (Request)

Повідомлення запрошення сервера клієнтів містить у першій строці: метод, який потрібно додати до ресурсу, ідентифікатор ресурсу та використовує версію протоколу.

Request = Request-Line *(general-header | request-header | entity-header) CRLF [message-body]

Рядок запиту(Request-Line).

Рядок запиту (Request-Line) починається з лексеми методу, потім слід запитуваний URI (Request-URI), версія протоколу і CRLF. Ці елементи поділяються SP. У рядку запиту (Request-Line) не припустимі CR і LF, виняток становить кінцева послідовність CRLF.

1.7 Метод (Method)

Мова методу вказує на те, який метод слід використовувати для обробки ресурсу, визначеного запитуваним URI (Request-URI). Цей метод враховує регістр символів.

Method = "OPTIONS" | "GET" | "HEAD" | "POST" | "PUT" | "DELETE" | "TRACE" | extension-method

extension-method = token

Список методів, які можна застосувати до ресурсу, може бути зазначений в полі заголовка Allow. Возвращает код стану відповіді завжди повідомляє клієнту, чи допустимо метод для ресурсу в даний час, так як набір допустимих методів може змінюватися динамічно. Серверів слід повернути код стану 405 (Метод не допустимо, Method Not Allowed), якщо метод відомий серверу, але не

застосуємо для запитаного ресурсу, і 501 (Не реалізовано, Not Implemented), якщо метод не розпізнано або не реалізований сервером. Список методів, відомих сервера, може бути зазначений в полі заголовка відповіді Public.

Методи GET і HEAD повинні підтримуватися всіма універсальними (general-purpose) серверами. Решта методи рекомендовані.

1.7.1 Поля заголовка запиту.

Поля заголовка запиту дозволяють клієнту передати серверу додаткову інформацію про запит і про самого клієнта. Ці поля діють як модифікатори запиту з семантикою, еквівалентній параметрами виклику методів в мовах програмування.

```
request-header = Accept | Accept-Charset | Accept-Encoding |
Accept-Language | Authorization | From | Host | If-Modified-
Since | If-Match | If-None-Match | If-Range | If-Unmodified-
Since | Max-Forwards | Proxy-Authorization | Range | Referer
| User-Agent
```

Безліч імен полів заголовка запиту (Request-header) може бути надійно розширено тільки в поєднанні зі зміною версії протоколу. Однак, нові або експериментальні поля заголовка можуть отримати семантику полів заголовка запиту (Request-header), якщо всі сторони з’єднання розпізнають їх як поля заголовка запиту (Request-header). Нерозпізнані поля заголовка обробляються як поля заголовка об’єкта (entity-header).

1.8 Відповідь(Response)

Після отримання та обробки запиту, сервер відправляє відповідне повідомлення HTTP-відповіді.

```
Response = Status-Line *( general-header | response-header
| entity-header ) CRLF [ message-body ]
```

1.8.1 Код статусу з описом

Статус-код (Status-Code) представляє собою трьохзначний цілочисельний код, який вказує на результат спроби зрозуміти та виконати запит. Пояснююча фраза (Reason-Phrase) призначена для короткого текстового опису коду стану. Код стану (Status-Code) призначений для використання автоматами.

Перша цифра в коді стану визначає клас відповіді, а останні дві цифри не мають особливого значення в класифікації. Існують 5 основних значень для першої цифри:

1. 1xx: Інформаційні коди - це означає, що запит було отримано, і триває обробка.
2. 2xx: Успішні коди - це вказує на те, що дія була успішно отримана, зрозуміла і оброблена.
3. 3xx: Коди перенаправлення - це вказує на те, що для виконання запиту потрібні подальші дії.
4. 4xx: Коди помилок клієнта - це вказує на те, що запит має неправильний синтаксис або не може бути виконаним з боку клієнта.
5. 5xx: Коди помилок сервера - це вказує на те, що сервер не в змозі правильно виконати запит.

1.8.2 Поля заголовків відповіді

Поля заголовка відповіді дозволяють серверу передавати додаткову інформацію про відповідь і ресурс, на який вона відноситься.

```
response-header = Age | Location | Proxy-Authenticate |
Public | Retry-After | Server | Vary | Warning | WWW-
Authenticate
```

Безліч імен полів заголовка відповіді (Response-header) може бути надійно розширити тільки в поєднанні зі зміною версії протоколу. Однак, нові або експериментальні поля заголовка можуть отримати семантику полів заголовка

відповіді (Response-header), якщо всі сторони з’єднання розпізнають їх як поля заголовка відповіді (Response-header). Нерозпізнані поля заголовка обробляються як поля заголовка об’єкта (entity-header).

1.9 Визначення методів

У HTTP/1.1 існує безліч загальних методів, наведених нижче. Незважаючи на те, що цей перелік може бути розширений, важливо зазначити, що додаткові методи можуть мати різну семантику, якщо вони використовуються різними клієнтами і серверами.

Поле заголовка Host повинно супроводжувати всі HTTP/1.1 запити.

1.9.1 Метод-GET

Метод GET дозволяє отримати інформацію, ідентифіковану запитуваною URI. Якщо ця URI вказує на процес, який генерує дані, сервер повинен повернути самі дані, а не вихідний текст процесу, якщо процес не генерує вихідний текст.

Існують умовний GET (conditional GET), при якому запит містить додаткові поля заголовка, такі як If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match або If-Range.

Також є частковий GET (partial GET), де запит містить поле заголовка Range. Цей метод дозволяє запитувати лише певну частину об'єкта, зменшуючи навантаження на мережу і дозволяючи отримувати лише необхідні дані.

1.9.2 Метод-POST

Метод POST актуальний при запиті, який адресується сервер приймає об’єкт, включений у запит, як нове підпорядкування (subordinate) ресурсу, ідентифікованого запитуваною URI (Request-URI) в рядку запиту (Request-Line). POST розроблений для реалізації таких функцій, як:

1. Анотація існуючих ресурсів;

2. Реєстрація повідомлення на електронній дошці оголошень (bulletin board), в конференції новин (newsgroup), списку розсилки (mailing list), або подібної групи статей;
3. Передача блоку даних, наприклад результат введення в формі, процесу обробки;
4. Розширення бази даних за допомогою конкатенуються операції (Append operation).

Фактично функція, виконувана методом POST, визначається сервером і звичайно залежить від запитуваної URI (Request-URI). Об'єкт, який передається методом POST, ставиться до цього URI так само як файл відноситься до каталогу, в якому він знаходиться, стаття відноситься до конференції новин (newsgroup), в якій вона зареєстрована, а запис відноситься до бази даних.

Дія, яку виконує метод POST, може або не повертати ресурс, ідентифікований URI, в результаті запиту. У випадку, коли відповідь не містить об'єкта, що описує результат, код стану відповіді може бути як 200 (OK), так і 204 (Немає вмісту, No Content).

Якщо ресурс був створений на основному сервері, відповідь повинна мати код стану 201 (Створено, Created) і включати об'єкт, який описує стан запиту та посилається на новий ресурс, разом з заголовком Location.

Відповідь з кодом стану 303 (Перенаправити, See Other) може бути використана для направлення агента користувача на кешований ресурс.

Запити POST повинні відповідати вимогам передачі повідомлення.

2 ОСНОВНІ ЗАГРОЗИ В МЕРЕЖІ ІНТЕРНЕТ

2.1 Аналіз основних загроз в мережі інтернет

Аналізуючи безпеку інформації в бездротових мережах, ми використовуємо відому модель безпеки, відому як тріада CIA (конфіденційність, цілісність, доступність), що застосовується до трьох сфер безпеки: інформаційної, інформаційної та кібернетичної. Наприклад, розглядаючи автоматизовану банківську систему, як інформаційний ресурс, ми розуміємо під інформаційною безпекою процес забезпечення конфіденційності, цілісності та доступності інформації для клієнтів банку.

У цій моделі інформаційна безпека означає забезпечення конфіденційності, цілісності та доступності інформації клієнтами банку на основі колективної та індивідуальної свідомості. Конфіденційність включає в себе забезпечення доступу до інформації тільки авторизованим користувачам, цілісність – гарантія відсутності модифікації інформації для користувачів, а доступність - забезпечення доступу до інформації авторизованими користувачами за необхідності.

Безпека інформації визначається як стан захищеності даних, де забезпечуються їх конфіденційність, доступність і цілісність, і вона включає в себе захист від ризику витоку інформації, несанкціонованого доступу та інших загроз в автоматизованій інформаційній системі.

Кібербезпека є набором засобів, стратегій та технологій, спрямованих на захист кіберсереди, ресурсів організацій та користувачів. Вона охоплює заходи для забезпечення безпеки ресурсів організацій і користувачів від кіберзагроз та включає в себе захист персональної інформації, виявлення, запобігання та реакцію на атаки. Стандарт ISO / IEC 27032 надає чітке розуміння зв'язку терміна "кібербезпека" з мережевою, прикладною та Інтернет-безпекою, а також безпекою критичних інформаційних інфраструктур.

Рисунок 2.1 – Взаємозв'язок між кібербезпекою та іншими доменами безпеки

Таким чином, для комплексних автоматизованих банківських систем можна використовувати відому модель тріади CIA, яка подається у формі на рис.2.2.

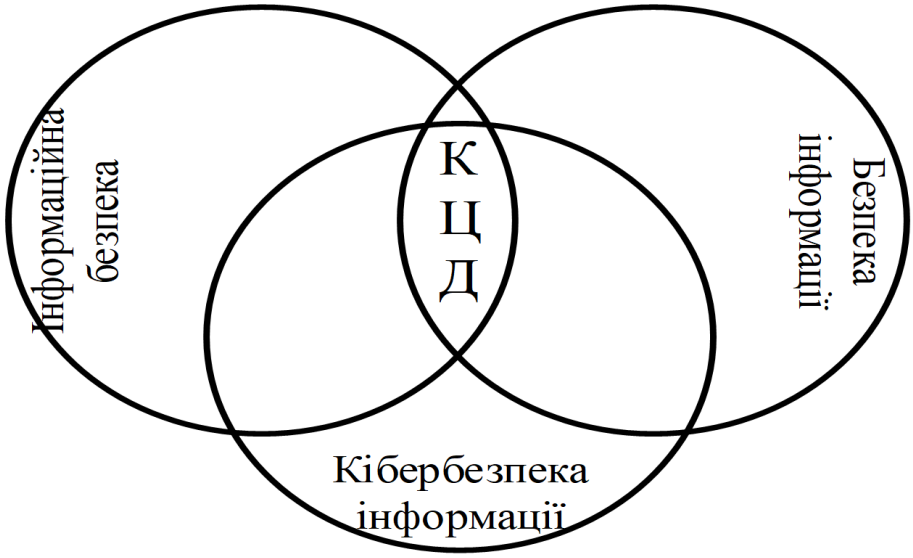

Рисунок 2.2 – Модель тріади CIA для комплексних АБС

Навіть при використанні різноманітних криптографічних алгоритмів на різних рівнях захисту, автоматизована банківська система вразлива перед різноманітними загрозами. Загальну класифікацію цих загроз можна знайти у трьох сферах безпеки рис. 2.3.

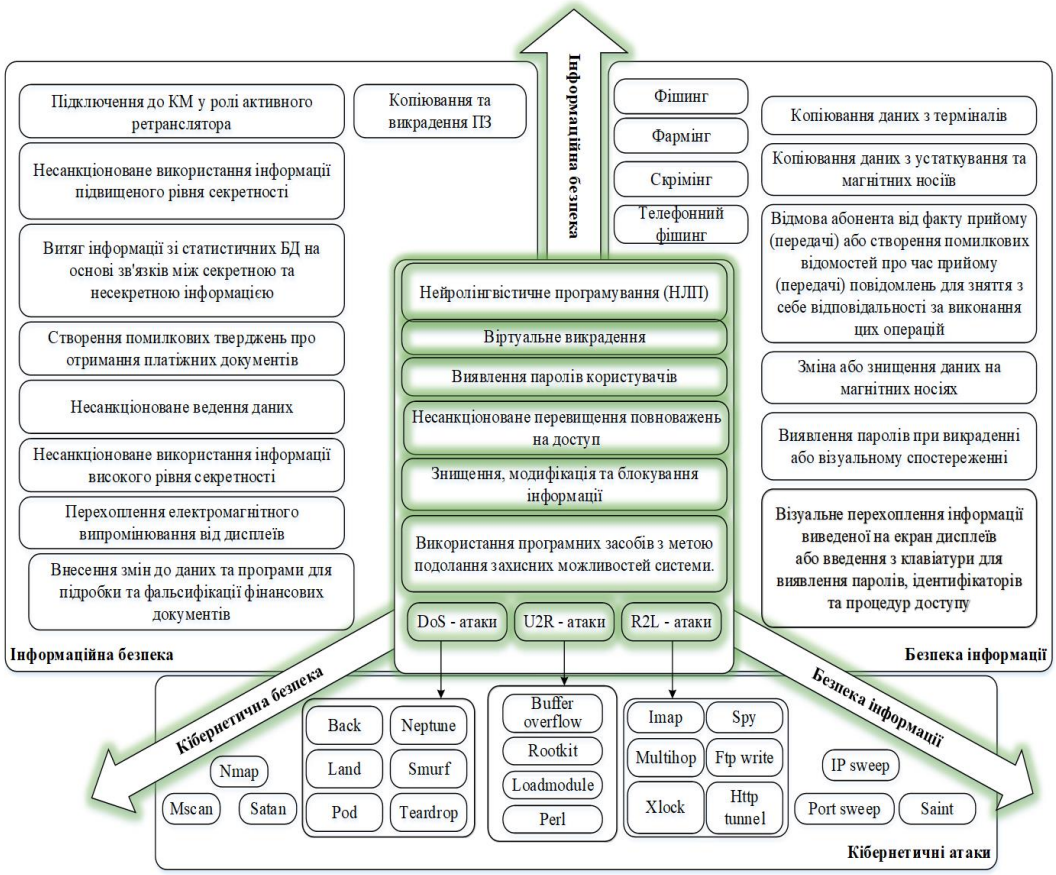

Рисунок 2.3 – Загальна класифікація загроз АБС

Загрози для банку представляють собою можливі або реальні дії зловмисників або конкурентів, спрямовані на завдання матеріальної чи моральної шкоди. Ці загрози поділяються за джерелами на внутрішні та зовнішні, а за спрямованістю та характером впливу на діяльність банків - на економічні, фізичні та інтелектуальні.

Економічні загрози включають корупцію, шахрайство, нечесну конкуренцію та використання неефективних технологій банківського виробництва, що може призвести до збитків або упущення вигоди для банку.

Фізичні загрози включають крадіжки, грабежі майна та коштів банків, поломки та виведення з ладу обладнання, що призводить до збитків та необхідності витрат на відновлення матеріальних ресурсів.

Інтелектуальні загрози включають розголошення або неправомірне використання банківської інформації, дискредитацію банку та соціальні

конфлікти, що може призвести до збитків, погіршення іміджу та соціальної напруженості.

Аналіз показав, що однією з найбільш вразливих областей в комплексних автоматизованих банківських системах є передача платіжних та інших повідомлень між банками, банкоматами та клієнтами. Це пов'язано з труднощами, такими як взаємне розпізнавання абонентів, захист електронних документів та процесу їх обміну, а також забезпечення виконання документів. Результати аналізу атак на комп'ютерні мережі компанії "Arbor Networks" підтверджують актуальність цих проблем (наведено на рис. 2.4).

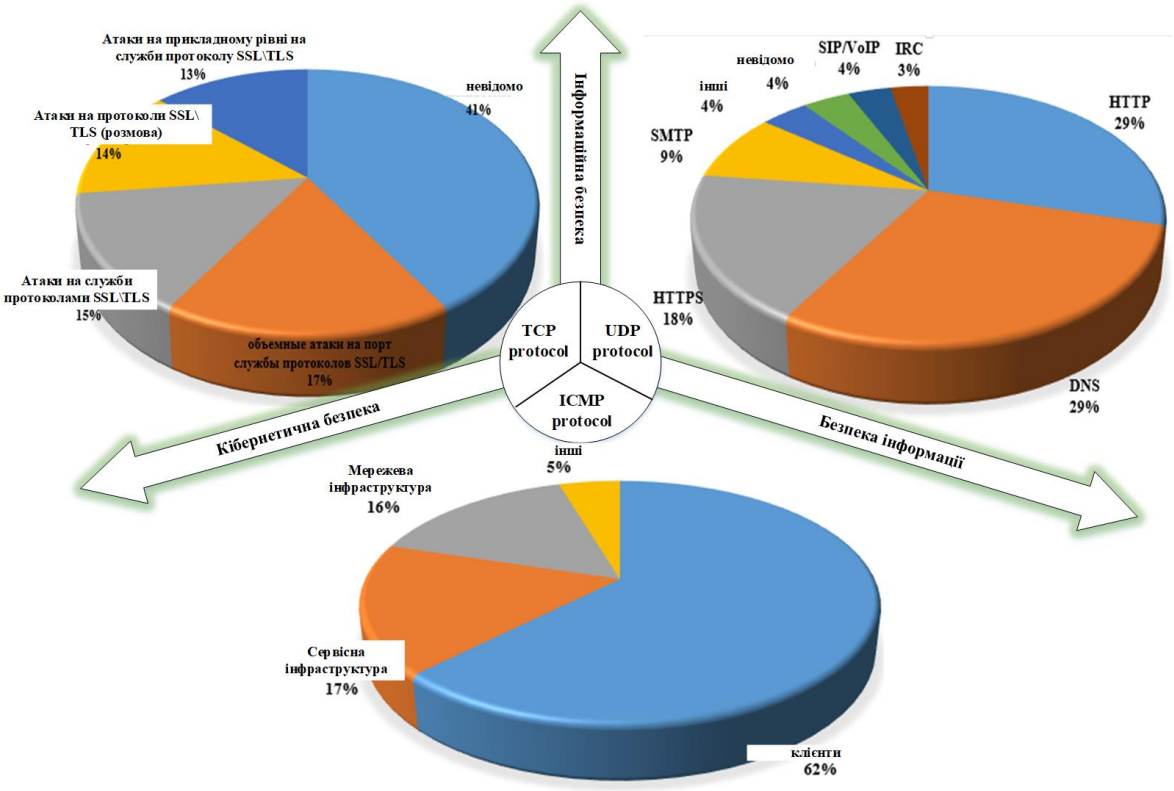

Рисунок 2.4 – Дослідження загроз на протоколи IP-мереж

Зі зростанням кіберзлочинності та розвитком обчислювальних можливостей зловмисників відбувається постійне удосконалення відомих кібератак і виникнення нових. Основний спосіб класифікації цих кібератак подано у вигляді схеми на рис. 2.5.

Атаки, спрямовані на проведення вторгнень, можна поділити на чотири категорії, кожна з яких включає різні типи атак, спрямованих на досягнення своєї мети вторгнення. Кожен окремий тип атаки становить загрозу мережі на відповідних рівнях мережевої моделі OSI та виконує свою функцію, спрямовану на завдання деструктивного впливу на мережу [4, 10, 13, 21, 24, 29]. До вказаних категорій атак відносяться такі:

Атаки типу DoS (відмова в обслуговуванні) - це мережеві атаки на комп'ютерну систему, які порушують властивість доступності інформації та блокують доступ до ресурсів користувачам. Ці атаки характеризуються перенавантаженням системи великою кількістю з'єднань, зловживанням ресурсами системи та викликанням помилок, пов'язаних із зміною параметрів конфігурації системи, що може призвести до перевантаження та збою в роботі серверів та комп’ютерних систем.

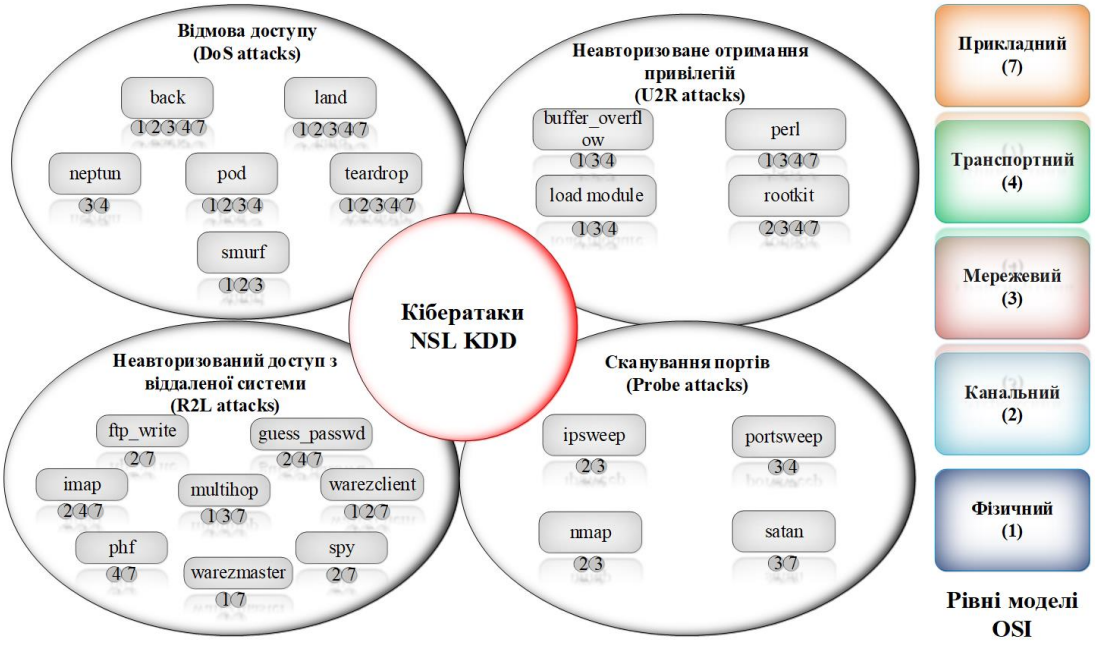

Рисунок 2.5 – Класифікація кібератак

Якщо атака відбувається одночасно з великої кількості IP-адрес, то таку атаку називають розподіленою (DDoS - Distributed Denial-of-Service). U2R атаки полягають у тому, що зловмисник отримує доступ до облікового запису

звичайного користувача та, використовуючи уразливість системи, незаконно отримує доступ до кореневого каталогу.

R2L атаки характеризуються тим, що незареєстрований користувач отримує доступ до мережі з боку віддаленої станції.

Probe-атаки включають в себе сканування мережевих портів з метою отримання конфіденційної інформації.

Вплив цих атак може бути розподілений за рівнями мережевої моделі OSI (табл. 2.1).

Таблиця 2.1 – Вплив атак на рівнях мережевої моделі OSI

Категорії атак	Типи атак	Рівні мережевої моделі OSI				
		Прикладний	Транспортний	Мережевий	Канальний	Фізичний
DoS	back	+	+	+	+	+
	land	+	+	+	+	+
	neptune		+	+		
	pod		+	+	+	+
	smurf			+	+	+
	teardrop	+	+	+	+	+
U2R	buffer_overflow		+	+		+
	loadmodule		+	+		+
	perl	+	+	+		+
	rootkit	+	+	+	+	
R2L	ftp_write	+			+	
	guess_passwd	+	+		+	
	imap	+	+		+	
	multihop	+		+		+
	phf	+	+			
	spy	+			+	
	warezclient	+			+	+
	warezmaster	+				+
Probe	ipsweep			+	+	
	nmap			+	+	
	portsweep		+	+		
	satan	+		+		

Кожен рівень моделі OSI підтримує конкретний набір мережевих протоколів (табл. 2.2).

Таблиця 2.2 – Мережеві протоколи різних рівнів моделі OSI

Рівень	Протоколи	Атаки	Приклад
Прикладний: доступ до мережевих служб	HTTP, gopher, Telnet, DNS, DHCP, SMTP, SNMP, CMIP, FTP, TFTP, SSH, IRC, AIM, NFS, NNTP, NTP, SNTP, XMPP, FTAM, APPC, X.500, AFP, LDAP, SIP, IETF, RTP,.	rootkit, back, land, teardrop, phf, perl, warezclient, imap, guess_passwd, warezmaster, ftp_write, spy, satan	Атаки відмови в обслуговуванні, розсилка спама електронною поштою
Транспортний: безпечне і надійне з'єднання "точка – точка"	ASP, ADSP, DLC, Named Pipes, NBT, NetBIOS, NWLink, Printer Access Protocol, Zone Information Protocol, SSL, TLS, SOCKS, PPTP	back, land, teardrop, imap, guess_passwd, pod, phf, buffer_overflow,perl load_module, rootkit, neptun, port_sweep	Атака SYN-пакетами (SYN Flood), атака ICMP-запитами зі зміненими адресами (Smurf Attack)
Мережевий: визначення маршруту і IP (логічна адресація)	TCP, UDP, NetBEUI, AEP, ATP, IL, NBP, RTMP, SMB, SPX, SCTP, DCCP, RTP, STP, TFTP	back, land, teardrop, satan, buffer_overflow, perl, load_module, rootkit, ip_sweep, nmap, neptun, port_sweep, smurf, pod	Атака ICMP-запитами (ICMP Flooding)
Канальний: MAC и LLC (фізична адресація)	IPv4, IPv6, ICMP, IGMP, IPX, NWLink, NetBEUI, DDP, IPSec, ARP, SKIP	back, land, teardrop, ftp_write, spy, imap, guess_passwd, warezclient, rootkit, ip_sweep, nmap, smurf, pod	Атака пакетами з різними MAC-адресами (MAC Flooding)
Фізичний: кабель, сигнали, бінарна передача	ARCnet, ATM, DTM, SLIP, SMDS, Ethernet, FDDI, Frame Relay, LocalTalk, Token Ring, PPP, PPPoE, StarLan, WiFi, PPTP, L2F, L2TP, PROFIBUS	back, land, teardrop, warezmaster, warezclient, buffer_overflow, load_module, smurf, pod	Атака спеціально сформованими пакетами (Dummy Packet Attack)

Отже, аналіз, який був проведений, підтверджує, що зростання кібератак є пропорційним еволюційному розвитку обчислювальної техніки та комп'ютерної грамотності зловмисників протягом останнього десятиліття. Висновки цього дослідження підкреслюють тісний зв'язок між розвитком технологій та зростанням загроз у сфері кібербезпеки.

2.2 Аналіз методів виявлення аномалій і зловживань.

У ході аналізу ризиків інформаційної безпеки можуть використовуватися спеціалізовані програмні рішення, які автоматизують процес обробки вихідних даних та обчислення значень ризиків.

Основою безпечної IT-інфраструктури автоматизованих банківських систем є тріада сервісів: конфіденційність, цілісність і доступність. Метою інформаційної безпеки є гарантування цих трьох основних сервісів безпеки, які відображають модель безпеки інформації, а саме конфіденційність, цілісність та доступність.

На рис. 2.6 наведені відомі моделі аналізу ризиків інформаційної безпеки.

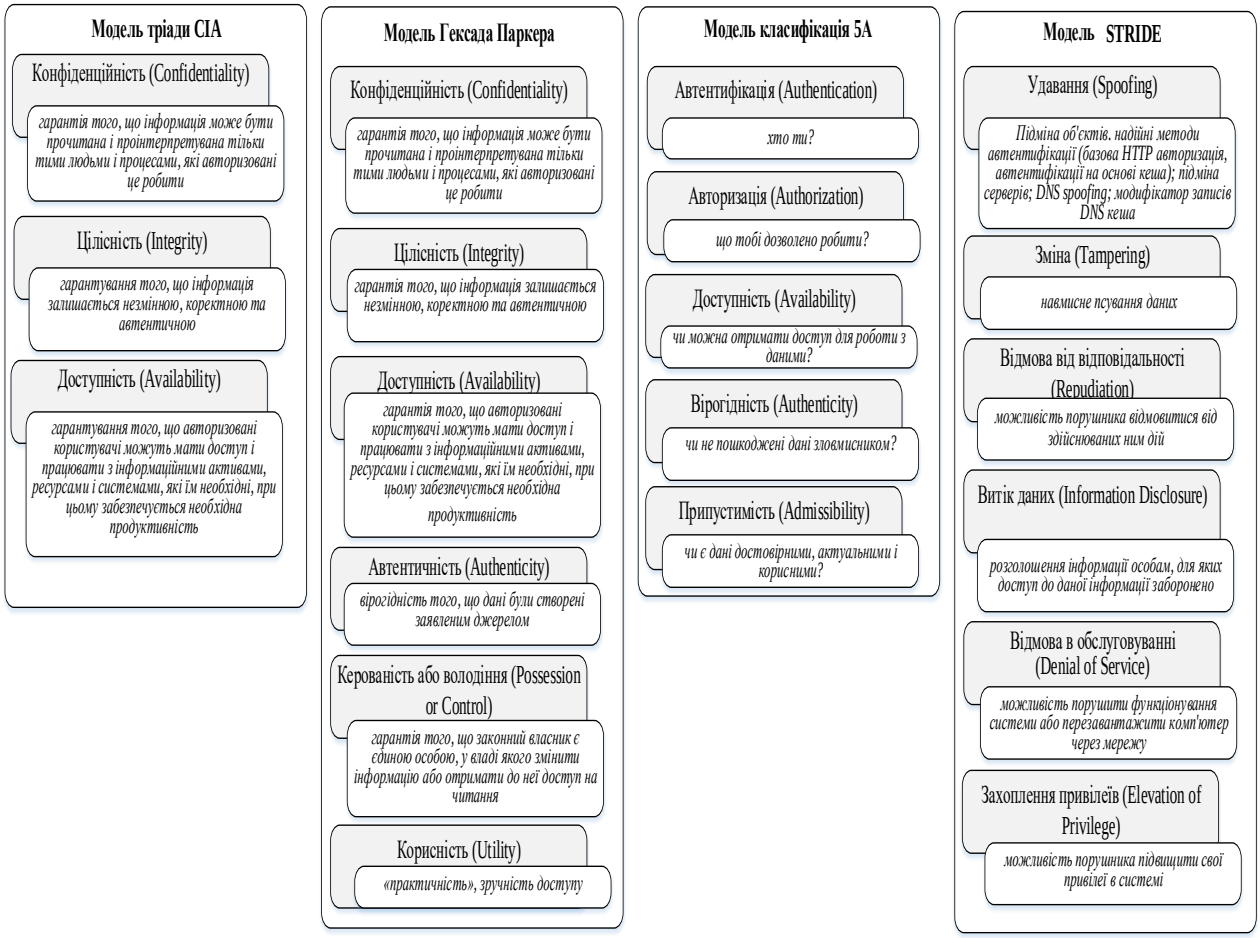

Рисунок 2.6 – Відомі моделі аналізу ризиків інформаційної безпеки

Зазвичай розрізняють дві основні групи методів оцінки ризиків безпеки. Перша група дозволяє визначити рівень ризику шляхом оцінки відповідності заданому комплекту вимог щодо забезпечення інформаційної безпеки. Друга група методик оцінки ризиків інформаційної безпеки ґрунтується на визначенні ймовірності реалізації атак і рівнів їхнього можливого збитку.

Метод виявлення атак вважається одним із ключових критеріїв оцінки систем захисту інформації. Основна класифікація методів представлена у вигляді схеми на рис. 2.7.

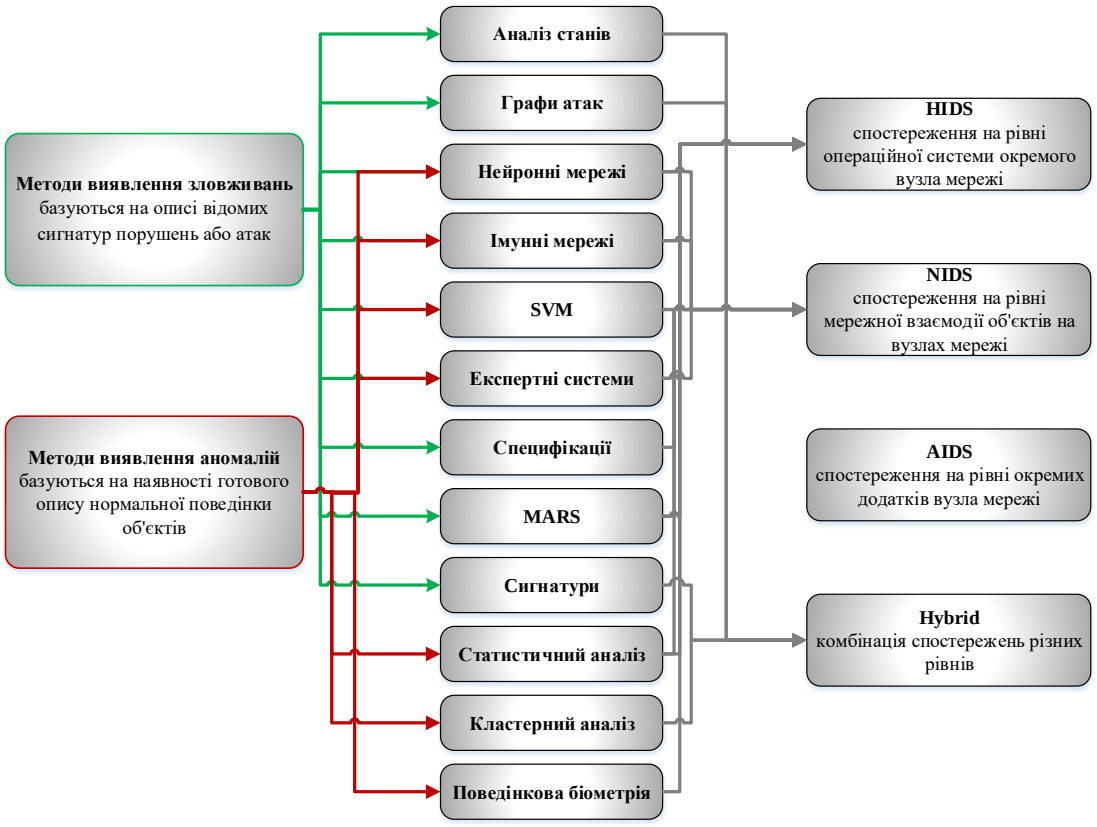

Рисунок 2.7 – Класифікація методів виявлення аномалій та зловживань

Основні характеристики методів представлені в табл. 2.3.

Таблиця 2.3 – Методи виявлення аномалій і зловживань

Метод	Вхідні дані	Математичний апарат	Опис	Вихідні дані	Економ. Ефективність	Обчислювальна складність
Аналіз систем станів	Шаблони нормальної поведінки системи, шаблони атаки	Теорія графів	Функціонування системи, що підлягає захисту, представляється через множину станів і множину переходів між ними.	Ймовірнісна оцінка реалізації атаки	Якісна оцінка	P
Графи сценаріїв атак	Модель системи, властивість коректності	Теорія графів	Множина поведінок ділиться на два класи - припустимі неприпустимі	Ймовірнісна оцінка реалізації атаки	Якісна оцінка	NP
Нейронні мережі	Траєкторії в деякому числовому просторі ознак	Алгоритми навчання нейронних мереж	Нейронні мережі навчаються на прикладах атак кожного класу надалі розпізнають приналежність поведінки одному з класів атак.	Ймовірнісна оцінка реалізації атаки	Якісна оцінка	P
Імунні мережі	Шаблони нормальної поведінки	Специфічні імунологічні теорії	Метод є механізмом класифікації і будується за аналогією з імунною системою живого організму.	Ймовірнісна оцінка реалізації атаки	Якісна оцінка	P
Support vector machines (SVM)	Вектори ознак нормальної поведінки системи, шаблони атаки	Алгоритми навчання і перенавчання	Метод подання та розпізнання шаблонів, який дозволяє формувати шаблони в результаті навчання, дозволяє обробляти вектори ознак великої розмірності.	Ймовірнісна оцінка реалізації атаки	Якісна оцінка	NP
Експертні системи	Факти про події в системі та правила виведення	Зіставлення фактів і правил	На підставі фактів і правил виводу система робить висновок про наявність чи відсутність атаки.	Ймовірнісна оцінка реалізації атаки	Якісна оцінка	NP
Заснований на специфікаціях	Специфікації атак	Аналіз даних	Невідповідність поведінки специфікації вважається атакою.	Ймовірнісна оцінка реалізації атаки	Якісна оцінка	NP
Сигнатурний	Події в системі, сигнатури атак	Аналіз даних	Методи працюють на найнижчому рівні абстракці, параметри системних викликів і записи файлів журналів.	Ймовірнісна оцінка реалізації атаки	Кількісна оцінка	NP

Аналіз систем виявлення аномалій (СВА) виявив основний недолік більшості сучасних комерційних СВА - їхню обмежену ефективність у виявленні

невідомих класів кібератак [10–12, 21, 26–35]. Зазвичай ці СВА використовують сигнатурні методи виявлення кібератак, що призводить до запізнення в організації процесу захисту.

2.3 Аналіз методик оцінки ризиків.

Узагальнюючи відомі методи оцінювання інформативності параметрів, представимо їх у формі схеми.

Рисунок 2.8 – Оцінювання інформативності параметрів

Вибір вищезазначених підходів та його адаптація для створення множини параметрів даних для системи захисту інформації повинна базуватися на одному або кількох критеріях якості [19, 28]. Щоб порівняти різні підходи за якістю є такі критерії: ступінь математичного обґрунтування підходу (Критерій 1); відносна складність реалізації підходу (Критерій 2); відносна швидкість процедур оцінювання інформативності параметрів (Критерій 3); якість параметрів, що оцінюється за Критерієм 4, діаграма розподілу показано на рис.2.9.

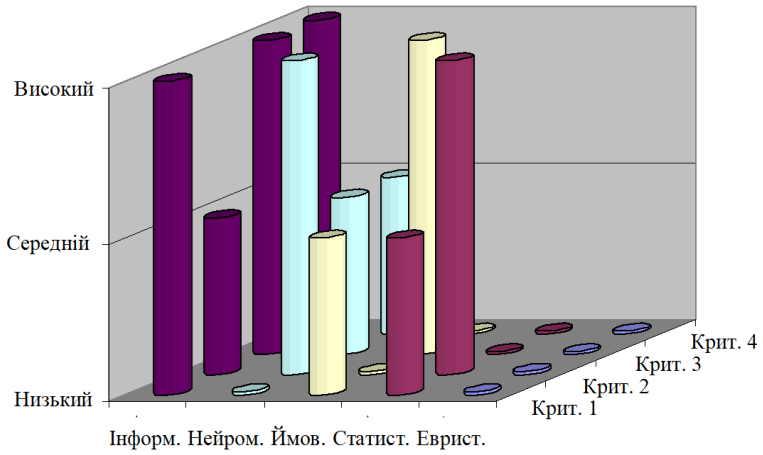

Рисунок 2.9 – Розпоідл критеріїв якості

Основуючись на проведеному аналізі [8, 9, 12–15, 17, 18], узагальнені результати порівняння ефективності розглянутих підходів наведені в табл. 2.4.

Таблиця 2.4 – Узагальнені дані за результатами оцінювання ефективності досліджуваних підходів

Підходи	Оцінка ефективності	
	кількісна	якісна
Інформаційний	0.22	достатня
Нейромережевий	0.36	достатня
Ймовірнісний	0.4	задовільна
Статистичний	0.4	задовільна
Евристичний	0.5	низька

Аналіз таблиці 2.4 показує, що серед розглянутих підходів інформаційний підхід є найефективнішим за рівних умов і може бути успішно використаний для формування інформативних параметрів для систем захисту інформації.

Оцінка ризиків інформаційної безпеки є ключовим та відповідальним етапом, оскільки її результати визначають подальші дії організації.

Метод оцінки ризиків представляє собою систематизовану послідовність дій, що дозволяє провести оцінку ризиків. Існують кількісні, якісні та комбіновані методи. Кількісні методи використовують об'єктивні дані для визначення вартості активів та імовірності втрат. Якісні методи використовують

рейтинг або категорії, такі як "низький", "середній", "високий", "не важливо" тощо. Комбіновані методи поєднують переваги обох підходів.

Оскільки загрози до банківських систем можуть мати різну природу, розглянемо деякі методики оцінки ризиків.

Таблиця 2.5 – Методики оцінки ризиків.

Мето-дика оцінки	Переваги	Недоліки	Підходи
1	2	3	4
NIST	- Детальний опис можливих ризиків інформаційних активів - Для підприємств різного розміру	- Довготривалий процес аналізу - Деякі функції не автоматизовано	Евристичний
FAIR	- Комплексний аналіз - Симуляційна модель - Висока ефективність	- Для крупних банків та підприємств	Ймовірнісний
IT-Grundshutz	- Гнучкість методу надає змогу проводити аналіз для будь-якої організації - Налаштовується на нові або існуючі активи	- Потребує теоретичної обізнаності процесу аналізу ризиків - Висока вартість ліцензії	Евристичний
OCTAVE	- Швидке впровадження - Обслуговує малі та середні за розміром підприємства	- Відсутність автоматизації - Не враховує специфіку банківської сфери	Евристичний
IRAM	- Відносна простота впровадження - Легкість в експлуатації менеджерами банківських установ	- Висока вартість ліцензії - Робота тільки з існуючими інформаційними активами	Інформаційний
EBIOS	- Велика кількість користувачів - Генерація звітів	- Лише для комерційних та державних установ	Інформаційний
RISK WATCH	- Простота впровадження та експлуатації - Гнучкість - Висока ефективність	- Аналіз ризиків лише на програмно-технічному рівні - Висока вартість ліцензії	Інформаційний
MEHARI	- Заснований на аналізі формул та параметрів - Формує оптимальну множину контрзаходів - У вільному доступі	- Застосовуваний до систем, що побудовані тільки за стандартом ISO	Евристичний
MAGERIT	- Систематичний метод аналізу - Кількісна оцінка - Гнучкість	- Результуючі дані залежать від людського фактору	Евристичний
CRAMM	- Детальне визначення існуючих ризиків - Ефективність використання	- Важкість у розумінні - Висока вартість ліцензії - Робота тільки з існуючими інформаційними активами	Ймовірнісний
Методика НБУ	- Детальний аналіз ресурсів банківської системи - Використання ризик-орієнтованого підходу	- Заснований на множині стандартів - Враховує специфіку лише українських банківських систем	Інформаційний

У табл. 2.6 наведені результати досліджень деяких методик оцінки ризиків.

Таблиця 2.6 – Результати досліджень методик оцінки ризиків

Методика	Атрибути							
	Якісна оцінка	Кількісна оцінка	Комплексна оцінка	Країна походження	Застосування у БС	Програмна реалізація	Ефективність контрзаходів	Простота розуміння
NIST	+			США	+	+	-	-
FAIR			+	США			+	+
EBIOS	+			Франція	+	+	+	-
MEHARI			+	Франція				
OCTAVE	+			США	+			
IT-GRUNDSHULTZ	+			Німеччина			+	
IRAM	+			Європа				+/-
RISK WATCH		+		США	+	+	+	+
FRAP	+			США				
CRAMM			+	Великобританія	+	+	+/-	+/-
MAGERIT	+	+		Іспанія	+	+		
Методика НБУ	+			Україна	+		-	+
Методика Корченко	+			Україна			+/-	+

Для отримання подальших оцінок еквівалентного грошового капіталу та непрямого відображення рівня захищеності рекомендується використовувати методики, які базуються на комплексному підході до оцінки ризиків. У цьому підході поєднані кількісні та якісні методи аналізу. До таких методик відносяться CRAMM і FAIR, структурні схеми яких представлені на рисунках 1.10 і 1.11

Рисунок 2.10 – Приклад застосування методика CRAMM

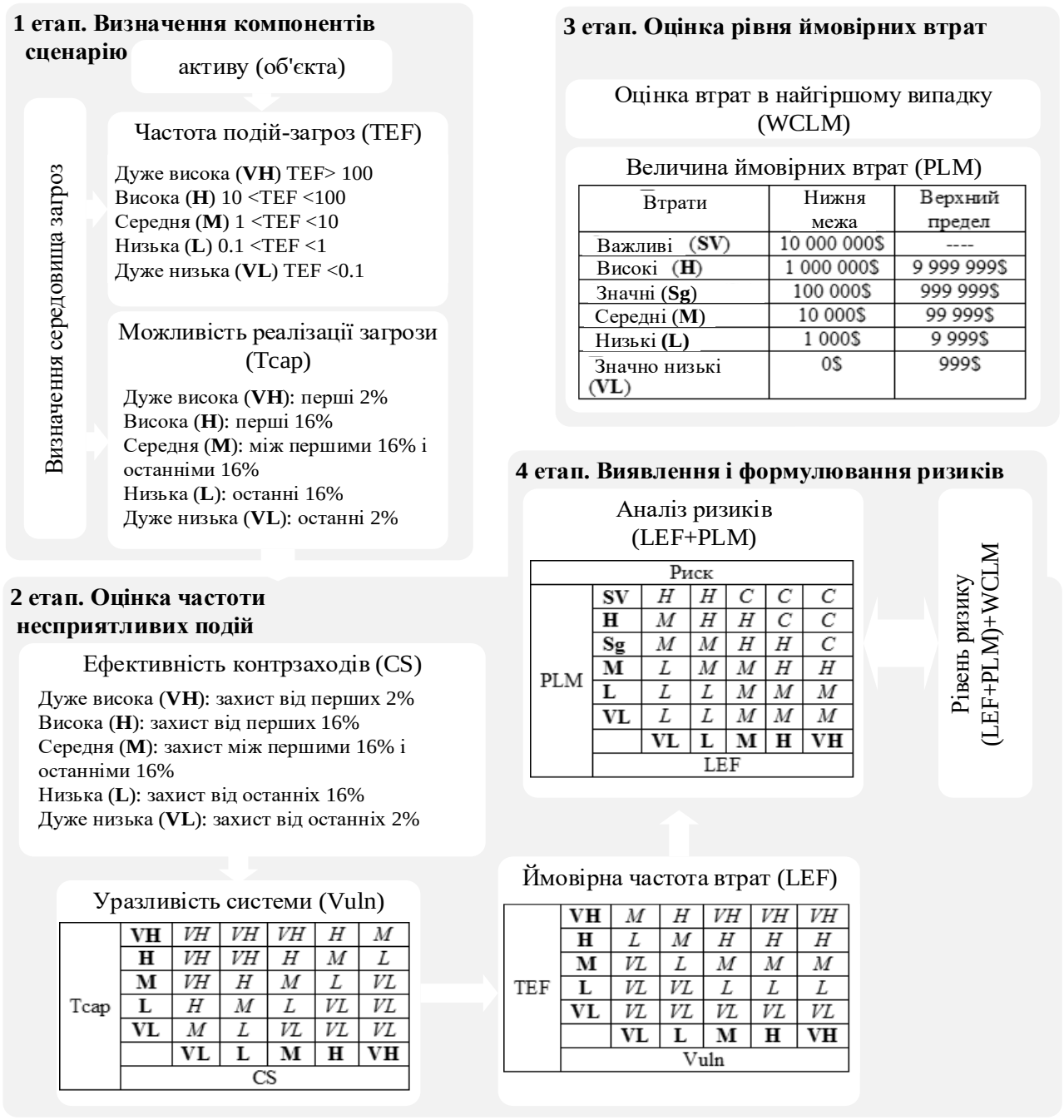

Рисунок 2.11 – Методика FAIR

Методи комплексного підходу до оцінки ризиків, як правило, включають наступні етапи [20, 21]:

1. Ідентифікація та визначення цінності ресурсів системи: аналіз меж досліджуваної системи, включаючи конфігурацію системи, інформацію про відповідальних осіб за фізичні і програмні ресурси, інформацію про користувачів системи та їх привілеї.

2. Ідентифікація ресурсів: визначення фізичних, програмних та інформаційних ресурсів, що знаходяться всередині кордонів системи; побудова моделі інформаційної безпеки системи (ІБ).

3. Оцінка загроз та вразливостей: ідентифікація загроз та оцінка рівнів загроз для груп ресурсів та їх вразливостей; оцінка залежностей між призначеними для користувачів сервісами та певними групами ресурсів; розрахунок рівнів ризиків і аналіз результатів. Замовник отримує ідентифіковані та оцінені рівні ризиків для своєї системи;

4. Пошук контрзаходів: пошук ефективних систем безпеки, що найкраще відповідають вимогам замовника; генерація кількох варіантів заходів протидії, адекватних виявленим ризикам та їх рівням.

Взаємозв’язок між методами виявлення атак та методиками оцінки ризиків представлено на рис. 2.12.

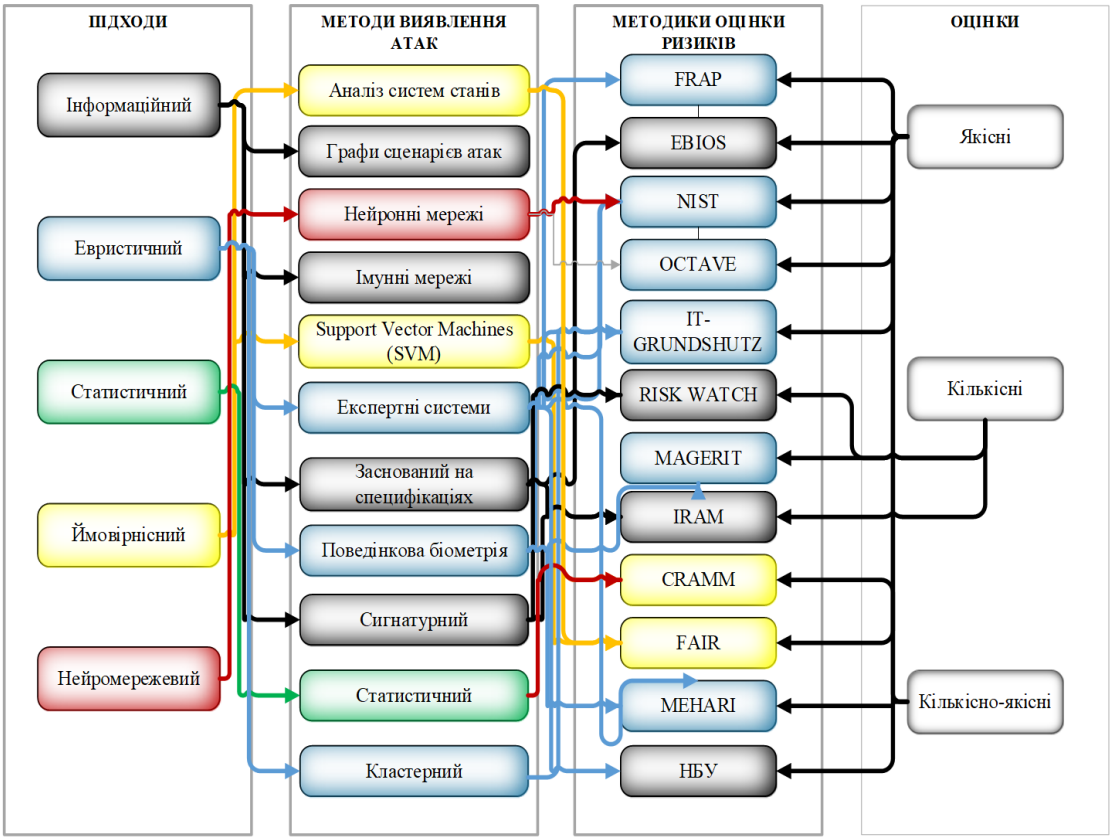

Рисунок 2.12 – Взаємозв'язок між методами виявлення атак та методиками оцінки ризиків

У зусиллях поліпшення продуктивності систем виявлення аномалій (СВА), обидва напрямки – розвиток існуючих та використання класифікаторів кібератак на основі дерев прийняття рішень – залишаються актуальними, несмотря на їхні переваги та обмеження. Ці напрямки продовжують активно розвиватися. Альтернативним варіантом є дослідження та подальший розвиток класифікаторів кібератак, які використовують дерева прийняття рішень. З правильною конструкцією цих класифікаторів можна досягти достовірних результатів класифікації при відносно низькій обчислювальній складності.

Важливу роль в процесі класифікації кібератак відіграють вхідні дані, які формують основу для створення класифікаторів СВА в комунікаційних системах. Використання загальновідомої бази даних KDD99, що включає близько 5 мільйонів класифікованих екземплярів атак, розділених на 22 типи і

охарактеризованих 41 ознакою, є доцільним для навчання і тестування цих класифікаторів.

Таблиця 2.7 – 41 ознака вектору мережевого з’єднання

№	Ознака	Опис
1	2	3
Основні ознаки		
1	Duration	Тривалість з’єднання (секунди)
2	Protocol_type	Тип протоколу (tcp, udp и др.)
3	Service	Мережева служба отримувача (http, telnet и др.)
4	Flag	Стан з’єднання
5	Src_bytes	Число байтів, переданих від джерела отримувачу
6	Dst_bytes	Число байтів, переданих від отримувача джерелу
7	Land	1 якщо з’єднання по ідентичних портах; 0 в інших випадках
8	Wrong_fragment	Кількість «невірних» пакетів
9	Urgent	Кількість пакетів з прапором URG
Ознаки, пов’язані із вмістом		
10	Hot	Кількість «hot» індикаторів, вміст яких: вхід до системної директорії, створення та виконання програм
11	Num_failed_logins	Кількість невдалих спроб входу
12	Logged_in	1 якщо успішний вхід; 0 в інших випадках
13	Num_compromised	Кількість вдалих спроб входу.
14	Root_shell	1 якщо досягнуто кореневої оболонки; 0 інших випадках
15	Su_attempted	1 якщо команда «su root» використовується; 0 в інших випадках
16	Num_root	Число підключень під «root» або число операцій, виконуваних від цього імені
17	Num_file_creations	Число операцій створення файлів у період з’єднання
18	Num_shells	Кількість shell повідомлень

Продовження таблиці 2.7

1	2	3
19	Num_access_files	Кількість операцій доступу до контрольних файлів
20	Num_outbound_cmds	Кількість вихідних команд у період FTP- сессії
21	Is_hot_login	1 якщо вхід виконано під «root» або «admin» правами; 0 в іншому випадку.
22	Is_guest_login	1 якщо гостьовий вхід; 0 в іншому випадку
Ознаки, пов'язані з часом		
23	Count	Кількість з'єднань між віддаленим та локальним хостами
24	Srv_count	Кількість підключень до локальної служби
25	Serror_rate	Відсоткове число з'єднань з помилкою типу SYN для даного хосту-джерела
26	Srv_serror_rate	Відсоткове число з'єднань з помилкою типу SYN для даної служби джерела
27	Rerror_rate	Відсоткове число з'єднань з помилкою типу REJ для даного хосту-джерела
28	Srv_rerror_rate	Відсоткове число з'єднань з помилкою типу REJ для даної служби джерела
29	Same_srv_rate	Відсоткове число підключень до служби
30	Diff_srv_rate	Відсоткове число підключень до різних служб
31	Srv_diff_host_ rate	Відсоткове число підключень до різних хостів
Ознаки, пов'язані з особливостями трафіку		
32	Dst_host_count	Кількість з'єднань з локальним хостом, встановлених віддаленою стороною
33	Dst_host_srv_ count	Кількість з'єднань з локальним хостом, встановлених віддаленою стороною, що використовують однакову службу
34	Dst_host_same _srv_rate	Відсоткове число підключень до локального хосту, встановлених відділеною стороною, що використовують однакову службу
35	Dst_host_diff_ srv_rate	Відсоткове число підключень до локального хосту, встановлених відділеною стороною, що використовують різні служби
36	Dst_host_same_src_port_rate	Відсоткове число підключень до даного хосту при поточному номері порту джерела
37	Dst_host_srv_ diff_host_rate	Відсоткове число підключень до служби різних хостів
38	Dst_host_serror_rate	Відсоткове число з'єднань з помилкою типу SYN для даного хосту-приймача
39	Dst_host_srv_serror_rate	Відсоткове число з'єднань з помилкою типу SYN для даної приймаючої служби
40	Dst_host_rerror_rate	Відсоткове число з'єднань з помилкою типу REJ для даного хосту-приймача
41	Dst_host_srv_rerror_rate	Відсоткове число з'єднань з помилкою типу REJ для даної приймаючої служби

Всі ознаки несуть різну ступінь інформативності. Розподіл рівня інформативності для мережевого з'єднання в залежності від кожної ознаки наведено у відсотковому виразі в табл. 2.8. Легко помітити, що ознаки, що визначають мережевий трафік, в структурному плані є зайвими.

Таблиця 2.8 – Відсотковий розподіл інформації про мережеве з’єднання згідно ознак

Ознака	1	2	3	4	5	6
% інформації	52,40	71,67	88,37	91,49	94,21	95,90
Ознака	7	8	9	10	11	12
% інформації	96,96	97,71	98,27	98,73	99,00	99,18
Ознака	13	14	45	16	17	18
% інформації	99,33	99,47	99,59	99,67	99,75	99,81
Ознака	19	20	21	22	23	24
% інформації	99,87	99,90	99,93	99,94	99,95	99,96
Ознака	25	26	27	28	29	30
% інформації	99,97	99,98	99,98	99,99	99,99	99,99
Ознака	31	32	33	34	35	36
% інформації	99,99	99,99	99,99	99,99	99,99	99,99
Ознака	37	38	39	40	41	
% інформації	99,99	100	100	100	100	

Враховуючи це, на сьогоднішній день рекомендується використовувати набір даних NSL-KDD, який є вдосконаленою версією набору даних KDD99. Цей набір не містить зайвих записів, 78% яких були присутні у KDD99. NSL-KDD спроектований для виявлення атак і використовує найбільш впливові 22 ознаки мережевого з'єднання.

Таблиця 2.9 – Вирішальні ознаки вектору мережевого з’єднання

№	Ознака
1	Duration
2	Protocol_type
3	Service
4	Flag
5	Source_bytes
6	Destination_types
7	Land
8	Wrong_fragment
9	Urgent
11	Failed_logins
13	Num_compromised
14	Root_shell
17	Num_file_creations
18	Num_shells
22	Is_guest_login
27	Rerror_rate
28	Srv_rerror_rate
29	Same_srv_rate
31	Srv_diff_host_rate
32	Dst_host_count
35	Dst_host_diff_srv_rate
37	Dst_host_srv_diff_host_rate

Для підтвердження ефективності використання набору даних NSL-KDD, було проведено експериментальне дослідження ступеня виявлення атак різних категорій. Результати цього дослідження наведені в таблицях 2.10–2.12.

Таблиця 2.10 – Результати виявлення DoS-атак

Back, %	Land, %	Neptune, %	Pod, %	Smurf, %	Teardrop, %
99,5	100,0	100,0	98,1	100,0	100,0
Середнє значення – 99,6 %					

Таблиця 2.11 – Результати виявлення R2L-атак

Ftp_write, %	Guess_passwd, %	Imap, %	Multihop, %
100,0	94,3	83,3	57,1
Warezclient, %	Warezmaster, %	Phf, %	Spy, %
65,0	90,0	100,0	100,0
Середнє значення – 86,2 %			

Таблиця 2.12 – Результати виявлення Probe-атак

Ipsweep, %	Nmap, %	Portsweep, %	Satan, %
65,2	100,0	99,9	99,3
Середнє значення – 91,1 %			

Як бачимо з таблиць найнижчим є відсоток виявлення R2L атак

3 СТВОРЕННЯ ПРОГРАМИ ДЛЯ ПІДВИЩЕННЯ РІВНЯ БЕЗПЕКИ ВЕБСЕРВЕРІВ “BAT”.

3.1 Опис предметної області програми “BAT”

Предметною областю дипломного проекту включає сферу запобігання викрадення будь-якої інформації із серверу. В процесі виконання дипломного проекту був реалізований застосунок для захищення даних користувачів на серверах .

За ідеологічну основу було взято функції протоколу HTTP , і сьогоднішні знання про те, що майже немає досить простої і водночас зрозумілої будь-якому користувачу програми , котра могла-б представляти інформацію швидко, безпечно і що найголовніше – безкоштовно. Більшість так званих програм аналізаторів (до якої відносится також мій застосунок) створенні для вузьконаправлених спеціалістів. Тому враховоючи те що сьогодні інформація – це найголовніше, я вважаю що повинен був створити застосунок котрий допоміг-би також людям котрі натрапляють на кіберзлочинців але не знають як з ними боротись.

Для користування , все що потрібно щоб користуватись програмою “BAT“ це лише інтернет зв’язок та дотримування рекомендацій описаних в допоміжному документі котрий буде отриманий в процесі установки програми “BAT” .

Сама програма знаходиться у відкритому доступі , на веб-сайті котрий був створений спеціально тільки для завантаження додатку. Після дотримування рекомендацій , користувач отримає готовий додаток з підсказками котрі завжди допоможуть отримати потрібну інформацію за допомогаю функції “Ключ-Значення” .

Як було зазначено зверху , основний механізм роботи пов’язаний із отриманням даних з документації RFC. Тобто майже всі ключі програми мають спільний механізм роботи. По-перше, необхідно отримати данні для порівняння надійності заголовків з тими що рекомендовані із тими що зараз

використовуються на сервері . По-друге треба відправити за допомогою спеціальної команди запит HEAD на адресу сайту, котрий буде цікавити користувача .

HEAD використовується потрібен лише для отримання відповіді від серверу без отримання інформації в тілі (відповідь POST) після чого залишається порівняти отримані данні (цей приклад використання був написаний , тому як я вважаю що зазначена функцій буде однією із головніших в щоденному використанні).

3.2 Огляд і аналіз існуючих аналогів, що реалізують функції предметної області

При аналізі аналогів програмного продукту, що розробляється, було виділено найпопулярніший online-сервіс «securityheaders.com» та «BAT»

В табл. 3.1, що наведена нижче, представлена відповідність базовим характеристикам, яким має відповідати “Аналізатор”

Таблиця 3.1 - Порівняльна характеристика продуктів-аналогів

Назва програмного продукту	«SH.com»	«BAT»
1	2	3
Функціональність		
USER_AGENT	-	+
Cookies (check)	-	+
X-XSS-Protection	+	+
HSTS	+	+
Перегляд заборгованостей	+	+
FeedBack	-	+
Except-CT	+	+
X-Permitted-Cross-Domain-Policies	+	+
Public-Key-Pins	+	+

X-Download-Options	+	+
FeedBack	-	+

«securityheaders.com»

Переваги:

- відгуки та рейтинги користувачів;
- багатоплатформовий;

Недоліки:

- доступ до сайту через браузер;
- нестабільна швидкість;
- інформація про сервер недостатньо детальна;

Онлайн сервіс «BAT».

Переваги:

- інтуітивно зрозумілий інтерфейс;
- детальна інформація про статус сервера;
- висока швидкість.

Недоліки:

- запуск тільки через Windows/Linux;
- відсутність рейтингової системи.

Розглянемо веб-додатки «securityheaders.com» й «BAT» більш детально. Головна сторінка с полем запису – це стандартний елемент будь-якого “Аналізатору”. Він , в наших двух випадку здійснюється через браузер та через безпосереднього запуску програми на комп’ютері здійснюється через електрону пошту. Для запису у випадку “SH.com” , потрібно знати лише URL-сайту , а для випадку з “BAT” , запуск програми (рис.3.1 – 3.2).

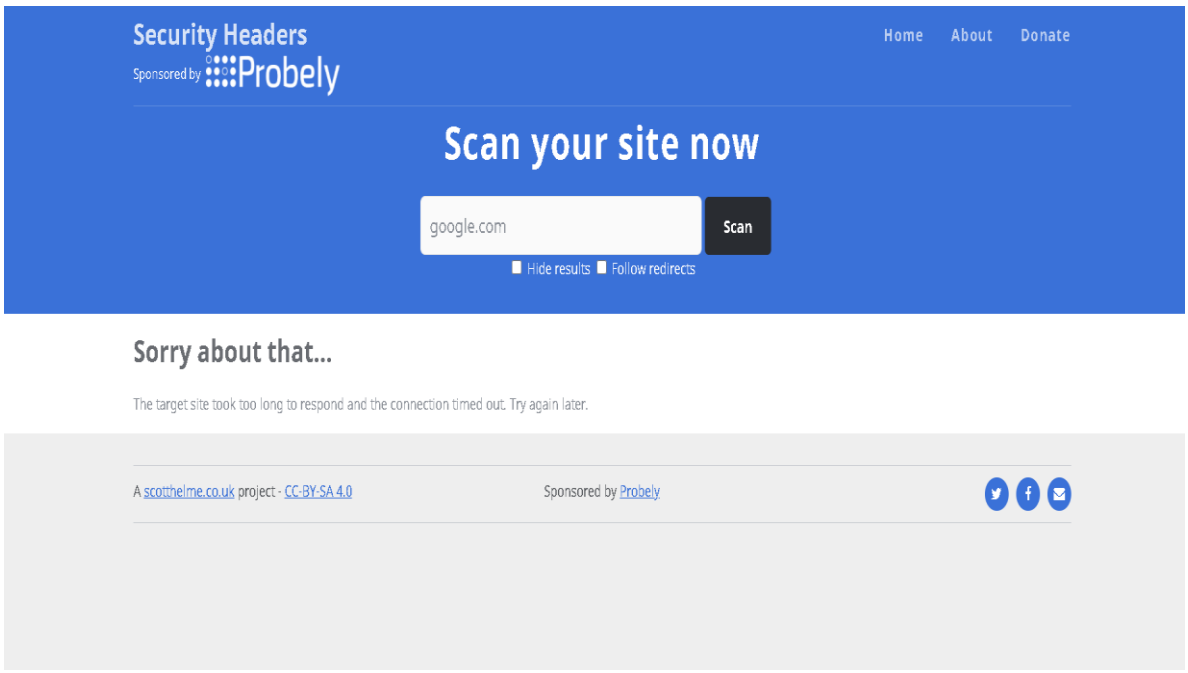

Рисунок 3.1 - Головна сторінка «SH.com»

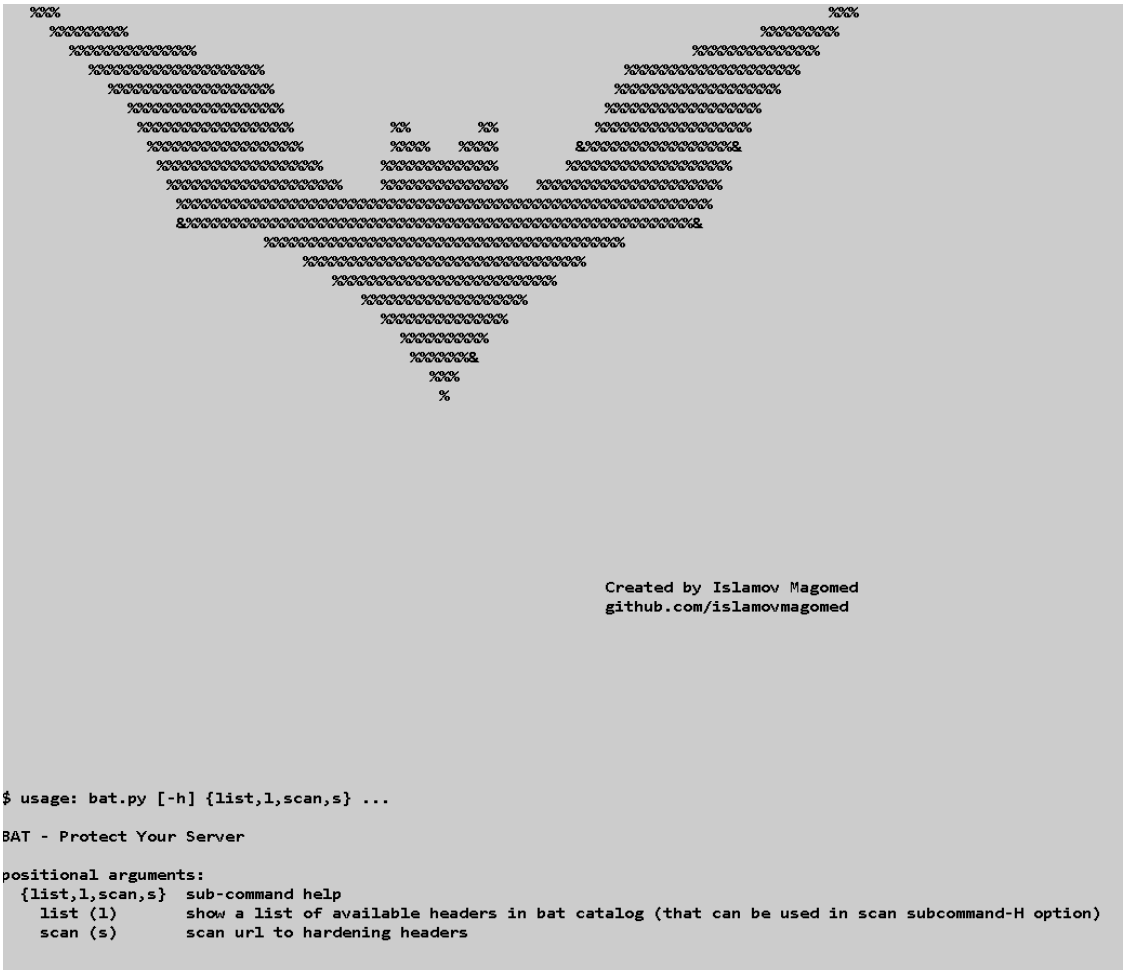

Рисунок 3.2 - Головне вікно «BAT»

Вхід до системи забезпечує користувачу можливість використання всіх можливостей додатку.

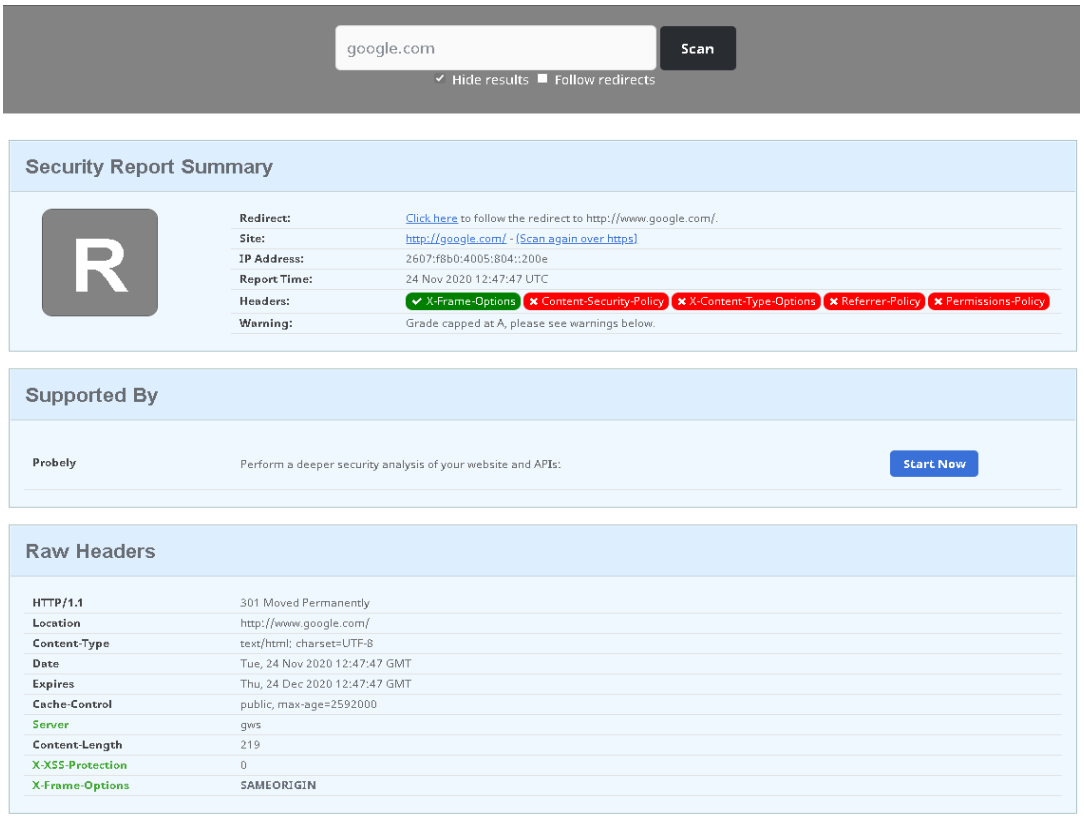

Рисунок 3.3 - Приклад виконання роботи «SH.com»

```
                                                    Created by Islamov Magomed
                                                    github.com/islamovmagomed




https://github.com/islamovmagomed/bat

Output explanation:
    [+] Good headers. Already used in your website. Good job!
    [+] Good headers. We recommend applying it
    [-] Bad headers. We recommend remove it


$ usage: bat.py [-h] {list,l,scan,s} ...

BAT - Protect Your Server

positional arguments:
  {list,l,scan,s}  sub-command help
    list (l)       show a list of available headers in bat catalog (that can be used in scan subcommand-H option)
    scan (s)       scan url to hardening headers

optional arguments:
  -h, --help       show this help message and exit
http://google.com
    [-] Cookie not only from SameSite
    [-] Server
    [+] Public-Key-Pins
    [+] Clear-Site-Data
    [+] Except-CT
    [+] X-Content-Type-Options
    [+] Feature-Policy
    [+] HTTP Strict Transport Security [HSTS]
    [+] Content-Security-Policy
    [+] X-Permitted-Cross-Domain-Policies
    [+] Referrer-Policy
    [+] X-Download-Options
```

Рисунок 3.4 - Приклад виконання роботи «BAT»

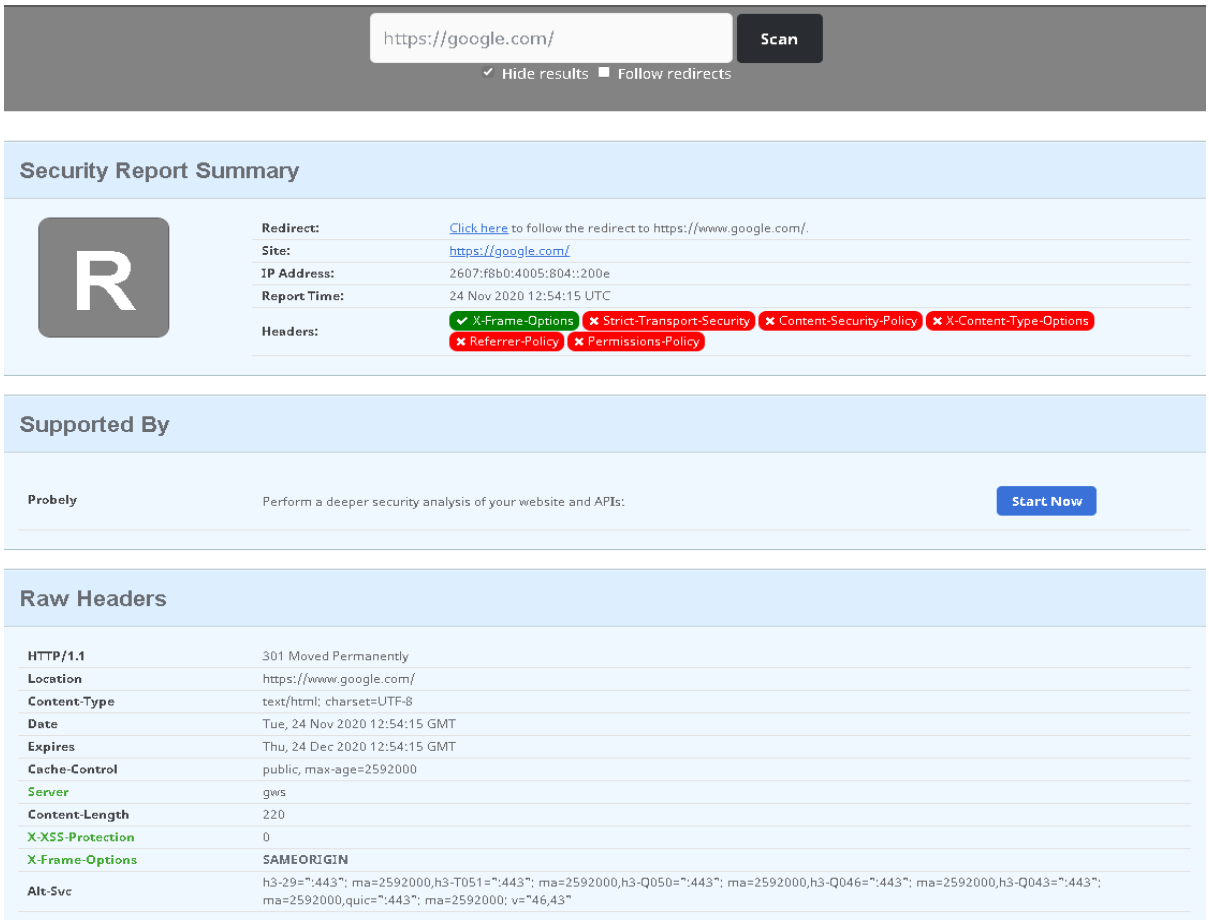

Рисунок 3.5 - Робота з функцією “don’t follow HTTP redirect”

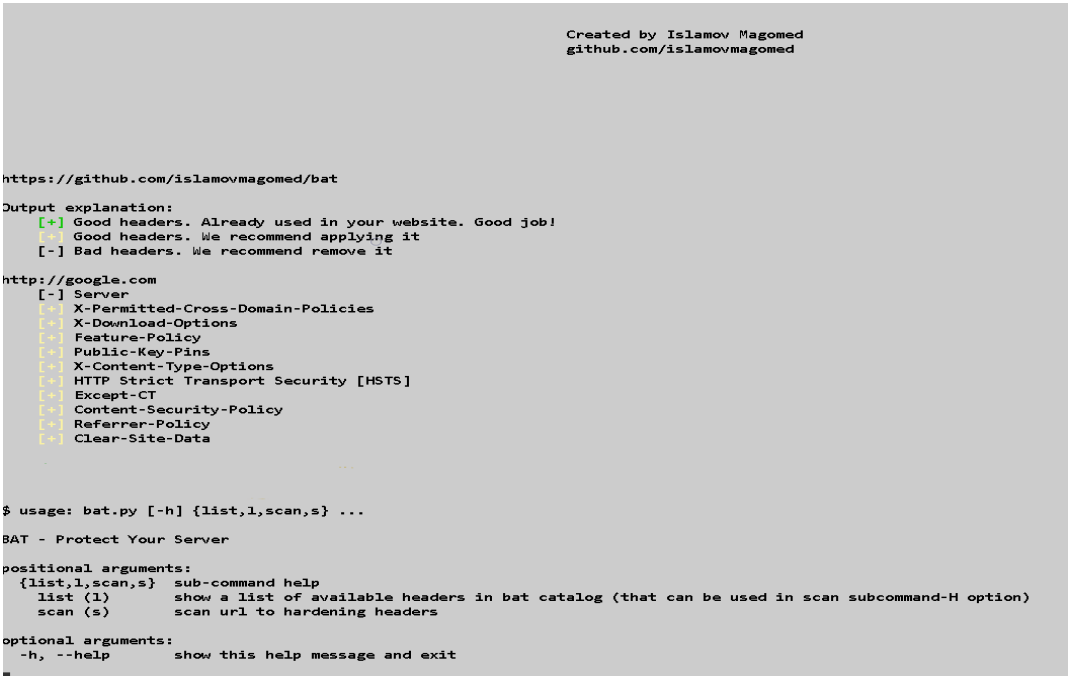

Рисунок 3.6 - Робота з функцією “don’t follow HTTP redirect”

3.3 Глосарій

Глосарій даної предметної області наведений в табл. 3.2.

Таблиця 3.2 - Глосарій предметної області

Термін	Опис терміну
1. Основні поняття и категорії предметної області	
BAT	Програмне забезпечення для підвищення безпеки вебсерверів.
2. Документи	
Bad.json	Вхідний документ, що являє собою форму реферальних посилань на джерела в яких міститься інформацій про загрози використання певних заголовків .
God.json	Вхідний документ, що являє собою форму реферальних посилань на джерела в яких міститься інформацій про безпечні у використанні заголовки.

3.4 Розроблення варіантів використання

Діаграма варіантів використання необхідна для формалізації функціональних вимог до системи та специфікації загальних особливостей поводження системи або сутності предметної області без розгляду внутрішньої структури. Будь-який з варіантів використання може бути підданий подальшій декомпозиції для утворення вихідної сутності. Декомпозиція проводиться залежно від функціональності та розгалуженість системи .

Діаграма варіантів використання усієї системи в цілому представлена в додатку Б. Модуль, що включає функцію GET-POST зображено на рис. 3.6.

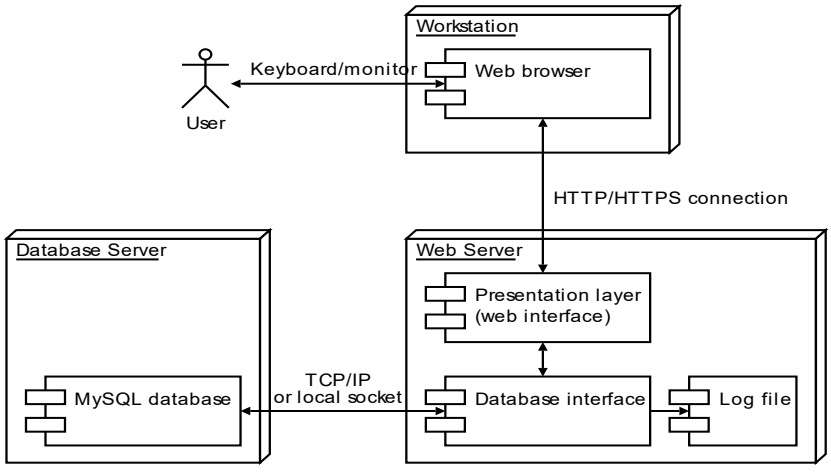

Рисунок 3.7 - Функція GET-POST

3.5 Специфікація варіантів використання

Опис основних варіантів використання наведено у табл. 3.3 – 3.4.

Таблиця 3.3 - Варіант використання «Scan»

Контекст використання	UC-01 Сканування серверу за допомогою запиту HEAD
1	2
Дійова особа	Користувач
Передумова	1.Завантаженя залежностей для успішного запуску програми. 2. Запуск програми 3. Введення команди. 4. Вхід
Тригер	Введення потрібних даних та натискання на кнопку «Enter»
Сценарій	1. Вхід у програму. 2. Заповнення поля для команди і запуск 3. Виведення інформації в форматі .cvs & .json. 4. Завантаження отриманої форми
Постумова	У випадку завершення програми , користувач отримує файл, який може бути використаний в програмному коді

Таблиця 3.4 - Варіант використання «Рекомендації»

Контекст використання	Перегляд світового стандарту безпеки для веб-серверів
Дійова особа	Користувач
Передумова	1.Завантаженя залежностей для успішного запуску програми.
Тригер	Запуск програми
Сценарій	1. Вхід у програму. 2. Заповнення поля для команди і запуск
Постумова	Отримання необхідної інформації для підвищення безпеки серверу.

3.6 Розкадровка варіантів використання

Розкадровка варіантів використання наведена відповідно розроблюваним модулям, а саме: модуль «Scan & Output in JSON-type» на рис. 3.7 та «Рекомендації» на мал. 3.8.

```
$ python bat.py s google.com -o json

Output explanation:
    [+] Good headers. Already used in your website. Good job!
    [+] Good headers. We recommend applying it
    [-] Bad headers. We recommend remove it


$ usage: bat.py [-h] {list,l,scan,s} ...

BAT - Protect Your Server

positional arguments:
  {list,l,scan,s}  sub-command help
    list (l)       show a list of available headers in bat catalog (that can be used in scan subcommand-H option)
    scan (s)       scan url to hardening headers

optional arguments:
  -h, --help       show this help message and exit
[
        {'url':  + url + ,
        'headers': [
                {"title": "Cookie not only from SameSite", "status": "toremove"},
                {"title": "Server", "status": "toremove"},
                {"title": "Feature-Policy", "status": "touse"},
                {"title": "Except-CT", "status": "touse"},
                {"title": "Public-Key-Pins", "status": "touse"},
                {"title": "Referrer-Policy", "status": "touse"},
                {"title": "Clear-Site-Data", "status": "touse"},
                {"title": "X-Content-Type-Options", "status": "touse"},
                {"title": "X-Permitted-Cross-Domain-Policies", "status": "touse"},
                {"title": "Content-Security-Policy", "status": "touse"},
                {"title": "HTTP Strict Transport Security [HSTS]", "status": "touse"},
                {"title": "X-Download-Options", "status": "touse"},
        ]}
]
```

Рисунок 3.8 - Результат команди “Scan & Output in JSON-type”

```
leo@davinci  /c/Program Files (x86)/BAT
$ usage: bat.py [-h] {list,l,scan,s} ...

BAT - Protect Your Server

positional arguments:
  {list,l,scan,s}  sub-command help
    list (l)       show a list of available headers in bat catalog (that can be used in scan subcommand-H option)
    scan (s)       scan url to hardening headers

optional arguments:
  -h, --help       show this help message and exit
http://hneu.edu.ua
    [-] Server
    [-] Cookie not over SSL/TLS
    [-] Cookie not only from SameSite
    [+] Except-CT
    [+] Content-Security-Policy
    [+] X-Frame-Options
    [+] X-XSS-Protection
    [+] X-Download-Options
    [+] X-Content-Type-Options
    [+] X-Permitted-Cross-Domain-Policies
    [+] Referrer-Policy
    [+] Clear-Site-Data
    [+] Feature-Policy
    [+] Public-Key-Pins
    [+] HTTP Strict Transport Security [HSTS]
```

Рисунок 3.9 - Результат команди “Recommendation”

3.7 Специфікація заголовків

Специфікація заголовків наведена в табл. 3.6.

Таблиця 3.6 - Функції програми “BAT”

Ідентифіка-заголовків	Назва заголовку	Атрибути вимог
		Виконавець
1	2	3
good	Clear-Site-Data	Ісламов М.В
good	Referrer-Policy	Ісламов М.В
good	HSTS	Ісламов М.В
good	X-Content-Type-Options	Ісламов М.В
good	X-Frame-Options	Ісламов М.В
good	Except-CT	Ісламов М.В
bad	Cookies not HTTP only	Ісламов М.В
bad	Cookies not over SLL/TLS	Ісламов М.В
bad	Cookies not only from SameSite	Ісламов М.В
bad	Content-Security-Policy to permisive	Ісламов М.В
bad	Public-Key-Pins without a back up key	Ісламов М.В
bad	Server	Ісламов М.В
bad	Access-Control-Allow-Origin too open	Ісламов М.В

3.8 Математична постановка задачі

При реалізації розроблюваного додатку всі функції реалізовані в один модуль

Створення програми типу “Аналізатор”

Створення логотипу

Створення сценарію завантаження програми

Створення веб-додатку для вільного розповсюдження програми

Розроблювана програма викладена в мережу Інтернет за назвою «BAT». Для переходу на сайт можна використати наступне посилання: bat.security. На головній сторінці сайту, маємо можливість побачити імітацію атаки – brute force, під якою знаходиться активне посилання на завантаження програми «BAT».

Після завантаження файлу з сайту (варто зазначити що сама програма «BAT» на цьому етапі буде знаходитись в архівному вигляді), клієнту буде потрібно роззархівувати архів.

Після чого залишиться лише запустити програму Setup.exe , яка необхідна для завантаження бібліотек для роботи виконавчого файлу BAT.exe .

Закінчивши процес установки , користувач отримує повноцінну програму-аналізатор

3.9 Створення файлів для зчитування інформації RFC

Головним етапом створення програми було створення файлів для отримання валідної інформації для роботи з серверами. Тобто іншими словами , треба було створити БД для роботи з отриманими даними зі сторони серверу. Вихідним результатом створення є структура бази даних та зв’язки між сутностями.

Та насамперед, перед початком розроблення, необхідно проаналізувати її вхідні/вихідні данні та виділити основні об’єкти даної області та задати індивідуальні атрибути, такі як, тип даних та обмеження.

3.10 Концептуальне інфологічне проєктування.

Інфологічний рівень представляє собою інформаційно-логічну модель (ІЛМ) предметної області, в якій відсутня надмірність даних, і відображені особливості об'єкта управління, враховуючи абстракцію від конкретних характеристик та особливостей СКБД.

Концептуальна модель відтворює логічну природу представлених даних, враження про дані з точки зору основних користувачів. Таким чином, у

концептуальній моделі основний акцент робиться на тому, що представлено в базі даних, а не на тому, як це представлено.

Інфологічна модель складається з набору атрибутів, що аналізуються та групуються для подальшого зберігання в БД . Об’єкти, виділені на фазі проектування, необхідно оформити в таблиці, що будуть зберігатись в даній БД та визначити зв'язки між ними. Для кожного поля існують деякі характеристики, що називаються атрибутами. Проєктування структури бази даних, а саме словник даних по предметній області, наведено в табл. 3.7. Інформація розписана для кожної таблиці окремо, для того, щоб покращити сприйняття інформації.

Реляційна схема даних, яка була розроблена, не вимагає подальшої нормалізації. Також, було враховано цілісність даних, тобто стійкість збережених даних до руйнування й знищення, що викликаються системними помилками й помилковими діями користувачів системи.

Таблиця 3.7 - Словник данних

№ п/п	Найменування Елемента	Тип і довжина	Призначення елемента
1	2	3	4
1. Таблиця «ChoiceBat»			
1.	result	varchar(20)	Отриманий результат
2.	content	int(20)	Підсказки
3.	category	varchar(20)	Значення для роботи с json файлами
2. Таблиця «ChoiceUser»			
1.	url	int 32	ідентифікатор
2.	status	BOOL	Коректність записаного посилання
3	List_parser	int(32)	Значення для отримана данних з клавіатури
4	Sub_parser	int(32)	Значення для отриманих дани з клавіатури
3. Таблиця «Summary»			
1.	Header_list	int 32	Дані які були зчитані з фалів .json
2.	Grou_output	varchar(40)	Позначення відповіді
3.	Total	int 32	Загальна отриманих результатів
4. Таблиця «Content»			
1.	jsonn	varchar(20)	Отримані данні для друку

2.	Cvs	varchar(20)	Отримані данні для друку

3.11. Створення глобальної інфологічної моделі даних.

Створення глобальної інфологічної моделі даних включає в себе об'єднання локальних інформаційних структур, які були отримані на попередньому етапі. При цьому об'єднанні локальних інформаційних структур утворюється глобальна модель за допомогою таких концепцій, як ідентичність, агрегація та узагальнення.

Ідентичність—однакове семантичне значення двох або більше об’єктів моделі.

Агрегація представляє собою абстракцію даних, яка дозволяє розглядати різноманітні об'єкти як єдиний новий об'єкт.

Узагальнення є абстракцією даних, яка дозволяє трактувати різні об'єкти одного класу як єдиний узагальнений тип об'єкта. Наприклад, об'єкти "scan_headers" і "scan_parsers" можуть належати до тієї самої категорії, що є прикладом ідентичності.

Зв'язки "check" і "show" можуть бути об'єднані в один зв'язок з новими атрибутами, що ілюструє, що при формуванні списків атрибутів документів та словників даних не всі терміни є синонімами.

3.12 Створення json файлів

Json представляє собою організацію даних, де структурні компоненти даних, їх складові та зв'язки між ними (незалежно від конкретного змісту чи середовища зберігання) ураховані в фізичній моделі даних.

3.12.1. Обґрунтування можливостей JSON файлів.

Основною вимогою до файлів даних яка використовується в BAT.py є оперативність виконання коректних та цілісних модифікацій у JSON. Час виконання транзакцій залежить від кількості даних в файлі, яка була розроблена для зберігання необхідної інформації.

Отже, серед позитивних аспектів JSON можна відзначити: його простоту та легкість сприйняття (оскільки використовується лише одна інформаційна конструкція - "таблиця"); жорсткі правила проектування, що базуються на математичних принципах; та повну незалежність даних.

Таблиця 3.8 - Обмеження унікальності

№ п/п	Атрибут або група атрибутів	Серед яких примірників, якої сутності або зв'язку має місце унікальність
1.	Files.read_json	Для всіх примірників сутності «ChoiceBat»
2.	Result	Для всіх примірників сутності «Summary»
3.	Arg.headers	Для всіх примірників сутності «ChoiceUser»
4.	Scan.parser	Для всіх примірників сутності «Content»

3.13 Тестування програмної системи

Для проведення повнофункціонального якісного тестування програмного продукту з боку працездатності , був розроблений окремий файл run.py , котрый надає змогу перевірити , чи були встановлені всі необхідні залежності на систему користувача .

Виведенням якості програмного забезпечення стало узагальнення результатів тестування, яке ґрунтується на перевірці протестованих функцій, виявленні дефектів та їх подальшому аналізі. Запропонований підхід включає в себе тестування навантаження, властивостей, інсталяції, регресії та графічного інтерфейсу користувача

Результати тестування програмного забезпечення наведено у табл. 3.9

Таблиця 3.9 - Перевірка роботи адмінської частини веб-додатку

№ п/п	Виконання тесту	Очікуваний результат	Отриманий результат	Проход-ження тесту
1	2	3	4	5
1.	Після запуску програми, отримати справку щодо шаблону виконання команд	Отримання підсказки котра дає можливість швидко зрозуміти як працювати з даним інструментом	Отримаємо результат с ключами для роботи с командами користувача	Тест пройдений
2.	Отримати вихідні дані в json-форматі	Отримання масиву даних в JSON	Отриманий масив	Тест пройдений

3.14 Розгортання програмного продукту

Розгортання додатку можливе на локальному та зовнішньому сервері.

Для розгортання додатку на ПК необхідно:

- частота процесора 2Ггц або більше;
- ОЗП не менше 1Гб;
- наявність підключення до мережі Internet;
- наявність Python3;
- наявність встановленої операційної системи Windows 7+ /Linux;

Також, був використаний Jenkins у якості Continues Integration, що дозволяє підхоплювати кожні зміни з git-репозиторію без жодних усиль й тримати стан серверу в найактуальнішому стані.

Jenkins цілком може забезпечувати задоволення максимальної кількості потреб і вирішення проблем, що виникають під час процесу розробки проектів різного рівня складності .

CI (Continuous integration system або система безперебійної інтеграції) — це система, яка дозволяє автоматизовувати певні процеси у розробці і контролі якості вашого проекту.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

У кваліфікаційній роботі магістра спроєктовано програмне забезпечення для аналізу та безпеки вебсерверів. Під час розв’язання задач дослідження, особливо практичної реалізації системи, враховано вимоги з охорони праці і техніки безпеки, пожежної та електробезпеки.

Виконання як теоретичної частини роботи, так і практичної, передбачає використання комп’ютерної техніки та обладнання з низькими напругами і силою струму. Зокрема, в якості блоку живлення плати ESP8266, використовувалась напруга живлення, яка становить 5 В. На платі використовуються можливі номінали напруги на рівні 5 В і 3,3 В, що не становить небезпеки для користувачів та розробника системи.

В якості регламентуючого документа з пожежної безпеки перед початком роботи над комп’ютерною системою для контролю параметрів мікроклімату теплиць використано вимоги «Типового положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України», які є діючим на даний час і затверджені постановою Кабінету міністрів України від 26 червня 2013 р. № 444.

Для організації захисту від негативного впливу екранів дотримано вимог Закону України "Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями" та НПАОП 0.00-7.15-18, який затверджений наказом Міністерства соціальної політики України 14.02.2018 N207. Робоче місце під час виконання кваліфікаційної роботи та проектування комп’ютерної системи облаштовано згідно наведених вимог та відповідає організаційним, ергономічним та вимогам з пожежної безпеки.

Електробезпеку робочого місця регламентують Правила безпечної експлуатації електроустановок споживачів, які затверджені наказом Держнаглядохоронпраці від 09.01.98 N 4, зареєстрованих у Міністерстві юстиції України 10.02.98 за N 93/2533 (НПАОП 40.1-1.21-98). Електромережа, яка

використовувалася при виконанні кваліфікаційної роботи магістра, відповідає правилам [23]:

– живлення електромережі проектовано, як окрему групову трьох провідну мережу з використанням фази, робочого «нуля» та захисного «нуля»;

– захисний «нуль» застосовано для реалізації заземлення електропристроїв;

– усі електричні та електронні пристрої мають захист від короткого замикання та непередбачуваних аварійних ситуацій;

– монтаж та експлуатація електромережі задовольняють вимогам щодо унеможливлення виникнення джерела загоряння через коротке замикання та перевантаження;

– усі лінії електроживлення виконанні не з легкозаймистого матеріалу або з негорючою ізоляцію;

– електричне устаткування підключено до мережі лише за допомогою справних штепсельних з’єднань і розеток заводського виготовлення;

– у розетках і штепселях передбачено контакти заземлення.

Вимоги електробезпеки при проектуванні компонентів комп’ютерної системи для контролю параметрів мікроклімату теплиць дотримано двома шляхами: використання безпровідних технологій передавання даних і напруги живлення в діапазоні 3,3В і 5 В, що дозволяє зменшити можливість ураження струмом при виникненні контакту з мережею чи в аварійних ситуаціях.

Щодо пожежної безпеки будівлі, де виконувався проект, то дотримано вимоги державних будівельних норм "Пожежна безпека об'єктів будівництва", які затверджені наказом Держбуду України від 03.12.2002 N 88, а також вимоги правил пожежної безпеки України, затвердженими наказом Міністерства України з питань надзвичайних ситуацій від 19.10.2004 N 126.

У приміщеннях, де розташовуються робочі місця користувачів ПК потрібно забезпечити відповідність вимогам санітарних норм і правил наведених у ДСанПіН 3.3.2-007-98 [24]. Крім цього, на робочих місцях, обладнаних комп’ютерами і периферійною технікою забезпечено оптимальні значення

параметрів мікроклімату: температури, руху повітря та відносної вологості, у відповідності до вимог нормативних документів.

Щодо освітлення, то приміщення де експлуатуються ПК, повинно бути обладнаним джерелами штучного освітлення та мати природне освітлення. Нормативний документ, який регламентує вимоги до рівнів природного і штучного освітлення – ДБН В.2.5-28-2018. Природне освітлення забезпечують прозорі вікна та інші світлові прорізи, що знаходяться на півночі або північному сході. У приміщеннях коефіцієнт природного освітлення повинен бути не нижче ніж 1,5 %. Розрахунок коефіцієнта природного освітлення виконують відповідно до методики, яка наведена у ДБН В.2.5-28-2018.

Штучне освітлення у приміщеннях з ПК забезпечується за допомогою системи загального освітлення, переважно рівномірного. В якості штучного джерела світла застосовуються люмінесцентні лампи типу ЛБ.

При використанні ПК для розробки проекту комп’ютерної системи для контролю параметрів мікроклімату теплиць на основі технологій інтернету речей було дотримано наступних вимог з техніки безпеки:

– не виконувався самостійний ремонт ПК і периферійних пристроїв;

– не вносились конструктивні чи інші зміни в апаратне забезпечення комп’ютера;

– використовувались тільки ті матеріали та предмети, які стосувались розробки комп’ютерної системи для контролю параметрів мікроклімату теплиць.

Для забезпечення вимог щодо безпечної експлуатації інформаційних технологій та мереж дотримано вимог СТУ EN 60950-1:2015 «Обладнання інформаційних технологій. Безпека. Частина 1. Загальні вимоги» (ДСТУ EN 60950- 1:2015).

4.2 Різновиди рятувальних робіт та надзвичайних ситуацій

Унаслідок надзвичайних ситуацій у населених пунктах країни і на підприємствах можуть виникнути руйнування, зараження місцевості радіоактивними та хімічними речовинами. Люди можуть опинитися у завалах,

пошкоджених та палаючих будинках, інших непередбачуваних ситуаціях. У зв'язку з цими обставинами буде потрібне проведення заходів із рятування людей, надання їм допомоги, локалізації аварій та усунення пошкоджень. При вирішенні цих проблем виходять з того, що в осередках ураження і районах лиха будуть проводитися не тільки суто рятувальні роботи, а й деякі невідкладні, що не пов'язані з рятуванням людей.

Рятувальні та інші невідкладні роботи (РіНР) проводяться з метою порятунку людей та надання допомоги ураженим, локалізації аварій та усунення пошкоджень, створення умов для наступного проведення відновлювальних робіт. При проведенні РіНР великого значення має дотримання певних умов.

Такими умовами є: своєчасне створення угруповань, сил, що залучаються для проведення РіНР, своєчасне ведення розвідки, швидкий рух і введення сил у осередок ураження, безперервне проведення РіНР до їх повного завершення, тверде й оперативне управління силами, що залучаються до проведення РіНР, і всебічне забезпечення їх діяльності.

Заходи, що відносяться до рятувальних робіт:

- розвідка маршрутів, за якими вводяться або виводяться формування ЦО;
- локалізація і гасіння пожеж;
- пошук і рятування людей з-під завалів;
- відкриття зруйнованих захисних споруд і рятування людей;
- подача повітря у завалені захисні споруди;
- надання ураженим першої медичної допомоги та їх евакуація;
- санобробка людей та знезараження їх одягу;
- знезараження місцевості, споруд, техніки.

Крім рятувальних робіт, в осередках ураження проводяться невідкладні роботи, до яких відносяться:

- прокладання маршрутних шляхів на заражених територіях і будування проїздів у завалах;

- локалізація аварій на комунально-енергетичних мережах, лініях зв'язку та їх відновлення;
- закріплення або ліквідація конструкцій споруд, які загрожують падінням та перешкоджають проведенню рятувальних робіт;
- ліквідація боєприпасів та інших вибухонебезпечних предметів (балони з газом, бочки з бензином тощо).

Керівництво проведенням усіх цих робіт у надзвичайних ситуаціях проводяться надзвичайними комісіями держави, області, міста тощо. При аваріях на об'єктах народного господарства, установах, якщо їх наслідки не виходять за межі об'єктів захисних зон, керівництво роботами проводиться адміністрацією підприємств.

Виникнення стихійних лих, а також аварій та катастроф можна в деяких випадках прогнозувати. Ці прогнози, як правило, закладаються в плани ЦО підприємств, установ, що передбачають попереджувальні заходи, які повинні зменшити наслідки аварій і катастроф.

Характер та обсяг таких заходів залежать від виду та рівня аварії або стихійного лиха, масштабів і часу їх виникнення. Загалом до таких заходів відносяться:

- приведення в готовність засобів захисту;
- перевірка готовності систем оповіщення;
- підготовка і видача населенню засобів індивідуального захисту та
- особистої профілактики;
- проведення санітарно-епідеміологічних заходів;
- підготовка до евакуації або відселення та їх проведення;
- вивезення матеріальних цінностей;
- захист продуктів харчування, джерел води тощо;
- герметизація приміщень і т.п.

Способи і послідовність проведення цих робіт залежать від обставин, що склались у районі аварії чи катастрофи, та наявності сил і засобів для проведення таких робіт.

Ліквідація наслідків надзвичайної ситуації проводиться для відновлення роботи підприємств, організацій, навчальних закладів тощо. При ліквідації наслідків надзвичайної ситуації здійснюються такі заходи:

- розвідка осередків надзвичайних ситуацій;
- локалізація і гасіння пожеж;
- відбудівля споруд і шляхів сполучення;
- проведення ізоляційне обмежених заходів в осередках інфекційного зараження;
- проведення спецобробки населення;
- дезактивація, дегазація техніки, майна, доріг, місцевості тощо.

Розвідку осередків надзвичайних ситуацій проводять сили Збройних сил, ЦО і невоєнізовані формування підприємств, організацій тощо. Воєнізовані сили розвідки ЗС і ЦО включають підрозділи радіаційної, хімічної, біологічної та інженерної розвідок.

У завдання цих підрозділів входить виявлення загального стану в осередках та визначення меж ураження, руйнування, повені І пожеж, а також виставлення спостереження на особливо важливих напрямках (станціях, переправах, перехресті доріг тощо). У місцях розташування евакуйованого населення, на маршрутах виходу з осередків надзвичайних ситуацій розвідка ведеться силами невоєнізованих формувань підприємств та організацій.

Аварійно-рятувальні й лікувально-евакуаційні заходи проводяться додатково до заходів, вжитих підрозділами ЗС, ЦО, медичних установ в осередках надзвичайних ситуацій. Ці роботи виконує населення, яке потрапило в осередок або розміщене на шляху розширення ураженого повітря, пожежі, повені тощо. Для допомоги у проведенні цих робіт в осередки надзвичайних ситуацій направляють сили і засоби спеціальних формувань ЗС, ЦО, Мінохорони здоров'я, комунальних служб, Міністерства охорони навколишнього середовища та інші.

Локалізація і гасіння пожеж проводяться з метою збереження матеріальних цінностей держави й окремих громадян. Здійснюється це протипожежними формуваннями ЗС, ЦО, МВС, Мінохорони навколишнього середовища із

залученням до цих робіт робітників, службовців і населення, що близько проживає до осередку надзвичайної ситуації. Відбудівля споруд і шляхів сполучення здійснюється з метою поновлення роботи життєво важливих органів міста, району тощо. До них належать: телеграф, телефон, лікарні, мости, залізниця, шляхи евакуації і підвезення матеріальних засобів та інші. Щоб запобігти поширенню епідемічних хвороб, проводять протиепідемічні заходи. До цих робіт залучають медичні заклади, санітарно дружини підприємств, навчальних закладів.

Усі протиепідемічні заходи в осередку організовує санепідемстанція або пересувний протиепідемічний загін. Проводять цю роботу медичні служби поліклінік, амбулаторій та інших лікувально-профілактичних заходів.

4.3 Сили і засоби для проведення рятувальних робіт

Для проведення рятувальних робіт залучаються невоєнізовані формування ЦО, військові частини і підрозділи, медорганізації тощо Невоєнізовані формування мають у першу чергу проводити рятувальні роботи на об'єктах народного господарства.

До них входять формування загального призначення, які мають у своєму складі аварійно-технічні формування, формування механізації робіт тощо. Вони можуть бути посилені протипожежними, дорожніми, автомобільними та іншими підрозділами. Від швидкості та рішучості дій формувань залежить життя багатьох людей та збереження матеріальних цінностей.

При рятувальних роботах потрібно дотримуватись таких заходів безпеки:

- пересування людей і автомобілів дозволяється тільки позначеними та розвіданими шляхами;
- забороняється вести роботи біля конструкцій, які загрожують падінням;
- зберігати режим радіаційного та хімічного захисту;
- проведення робіт у задимлених та загазованих приміщеннях групами по 2—3 чоловіки в особистих засобах захисту;

- проведення робіт на електромережах, електроустановках тільки після їх відключення і заземлення;
- освітлення ділянок роботи вночі та за несприятливої погоди.

Під стійкістю роботи об'єкта народного господарства розуміють здатність підприємства, установи попереджувати виникнення виробничих аварій, катастроф, протистояти впливу уражаючих факторів, аби запобігти або зменшити загрозу життю і здоров'ю робітників і службовців, матеріальних втрат, а також забезпечити відновлення порушеного виробництва в мінімально короткий термін.

Стійка робота промислового підприємства складається:

- зі стійкості інженерно-технічного комплексу (будівель, споруд, систем енерго-, газо-, водозабезпечення тощо) до дії зовнішніх факторів при аваріях, катастрофах, а також при застосуванні щодо них сучасної зброї;
- зі стійкості виробничої діяльності (захист виробничого персоналу, надійність систем управління, поновлення роботи у стислі терміни).

Фактори, від яких залежить стійка робота об'єктів у НС мирного і воєнного часу:

- Надійність захисту робітників і службовців.
- Безпечність розташування об'єкта щодо зон масштабних зруйнувань.
- Можливість інженерно-технічного комплексу протистояти уражаючим діям сучасної військової зброї.
- Безперервність постачання електроенергією, паливом, газом і всім необхідним для випуску продукції.
- Надійність керування виробництвом, силами і засобами ЦО.
- Підготовленість підприємства до поновлення виробництва і проведення рятувальних робіт.

ВИСНОВКИ

Програмне забезпечення “BAT”, про яке йшла мова в цій кваліфікаційній роботі була створена на замовлення компанії “SoftServe”, в часи коли компанія отримала великі грошові збитки від ефективно реалізованої кібератаки причиною якої став “фактор людини”.

Зіпсування репутації призвело до різкого падіння на рикну цінних паперів що в свою чергу призвело до неминучих кадрових рішень, звільнень . Тобто на даному прикладі можна засвідчитись що безпека кіберпростіру не в майбутньому а уже зараз має бути поставлена на головні позиції IT-індустрії.

Функціональність додатку “BAT” полягає в забезпеченні зручного та швидкого перегляду статусу вебсерверу , знаходженя слабких мість які в свою чергу , можуть використати кіберзлочинці

При виконанні кваліфікаційної роботи використовувалися такі інструментальні засоби: Python, Bash, Vim, Nsi .

Документ включає усі вхідні та вихідні документа та опис проведеного тестування розробленого модулю.

В подальшому планується удосконалення програмного продукту з метою реалізації додаткових функціональностей та оптимізаціїї програмного продукту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бурячок В. Л. Політика інформаційної безпеки [Текст] : підручник / В. Л. Бурячок, Р. В. Грищук, В. О. Хорошко ; під заг. ред. проф. В. О. Хорошка. – К. : ПВП «Задруга», 2014. – 222 с.

2. Report on Post-Quantum Cryptography, [Электронный ресурс]. URL: http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf (Дата обращения 255.12.2019).

3. GOST 34.310-95. Informacionnaja tehnologija. Kriptograficheskaja zashhita informacii. Procedura vyrabotki i proverki jelektronnoj cifrovoj podpisi na baze asimmetrichnogo kriptograficheskogo algoritma [Tekst]. – K.: Gosstandart Ukrainy, 1998.

4. GOST 34.311-95. Informacionnaja tehnologija. Kriptograficheskaja zashhita informacii. Funkcija heshirovanija [Tekst]. – K.: Gosstandart Ukrainy, 1998.

5. GOST R34.10-94. Informacionnaja tehnologija. Kriptograficheskaja zashhita informacii. Procedury vyrabotki i proverki jelektronnoj cifrovoj podpisi na baze asimmetrichnogo kriptograficheskogo algoritma [Tekst]. – Nacional'nyj standart

6. Edited by Serhii Yevseiev, Volodymir Ponomarenko, Oleksandr Laptiev, Oleksandr Milov. Synergy of building cybersecurity systems: monograph / S. Yevseiev, V. Ponomarenko, O. Laptiev, O. Milov and others. – Kharkiv: PC TECHNOLOGY CENTER, 2021. – 188 p.

7. ДСТУ 4145–2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка [Текст]. – К.: Держстандарт України, 2002. – 40 с.

8. ДСТУ 7564–2014. Інформаційні технології. Криптографічний захист інформації. Функція ґешування [Текст]. – К. : Держстандарт України, 2014. – 39 с.

9. ДСТУ 7624–2014. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення [Текст]. – К.: Держстандарт України, 2014. – 235 с.

10. Modeling of security systems for critical infrastructure facilities: monograph / S. Yevseiev, R. Hryshchuk, K. Molodetska, M. Nazarkevych and others. – Kharkiv: PC TECHNOLOGY CENTER, 2022. – 196 p.

11. Models of socio-cyber-physical systems security: monograph / S. Yevseiev, Yu. Khokhlachova, S. Ostapov, O. Laptiev and others. – Kharkiv: PC TECHNOLOGY CENTER, 2023. – 168 p

12. Звід правил для управління інформаційною безпекою (ISO/IEC 27002:2005, MOD): СОУ Н НБУ 65.1 СУІБ 1.0:2010 [Текст]. – К.: НБУ, 2010. – 209 с.

13. Корченко А. А. Банківська безпека [Текст] / А. А. Корченко Л. Н. Скачек В. А. Хорошко. – Київ, 2014. – 185 с.

14. Стандарт України СОУ Н НБУ 65.1 СУІБ 1.0:2010. Методи захисту в банківській діяльності система управління інформаційною безпекою. Вимоги. (ISO/IEC 27001:2005, MOD) [Текст]. – К.: НБУ., 2010. – 67 с.

15. S. Yevseiev and other “Development of the interacting agents behavior scenario in the cyber security system”. Eastern-European Journal of Enterprise Technologies. 2019. 5/9 (101). p. 46–57.

16. Security of Internet Banking - A Comparative Study of Security Risks and Legal Protection in Internet Banking in Thailand and Germany [Electronic resource]. – Available at: http://www.thailawforum.com/articles/internet-banking-thailand.html. Accessed on: Des. 09, 2019.

17. Методичні рекомендації щодо впровадження системи управління інформаційною безпекою та методики оцінки ризиків відповідно до стандартів національного банку України/ [Електронний ресурс]. – Режим доступу: zakon.rada.gov.ua/laws/show/v0365500-11

18. Міжбанківські розрахунки в Україні [Електронний ресурс]. – Режим доступу: http://www.bank.gov.ua/control/uk/publish/. Accessed on: Des. 09, 2019.

19. Основи створення комплексної системи економічної безпеки підприємства: теоретичний аспект [Електронний ресурс] / Коваленко К.В. – Режим доступу до статті http://www.nbuv.gov.ua. Accessed on: Des. 09, 2019.

20. Національний Банк України. Платіжна організація національної системи масових електронних платежів. [Електронний ресурс]. – Режим доступу: http://zakon5.rada.gov.ua/laws/show/va119500-08. Accessed on: Des. 09, 2019.

21. Yevseiev and other “Development of a scenario modeling of conflicts tools in a security system based on formal grammars”. Eastern-European Journal of Enterprise Technologies . 2019. 6/9(102). p. 22–39.

22. Стандарт України СОУ Н НБУ 65.1 СУІБ 1.0:2010. Методи захисту в банківській діяльності система управління інформаційною безпекою. Вимоги. (ISO/IEC 27001:2005, MOD) [Текст]. – К.: НБУ., 2010. – 67 с.

23. Міжбанківські розрахунки в Україні [Електронний ресурс]. – Режим доступу: http://www.bank.gov.ua/control/uk/publish/.

24. Старинський М. В. Щодо визначення поняття “банківська інформація” та виділення її видів / [Електронний ресурс]. – Режим доступу: uabs.edu.ua/images/.../K.../Starinskii_s_015.pdf. Accessed on: Des. 09, 2019.

25. Stanislav Milevskyi, Volodymyr Aleksiyev, Olha Korol, Oleksandr Milov, and Serhii Yevseiev. Security Analysis Models for Multimedia Information Resources in Social Networks. Cybersecurity Providing in Information and Telecommunication Systems, January 28, 2021, Kyiv, Ukraine. P. 60–67.

26. ISO/IEC 27032:2012 Information technology – Security techniques – Guidelines for cybersecurity [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44375. Accessed on: Des. 09, 2019.

27. НАЦІОНАЛЬНИЙ БАНК УКРАЇНИ. Платіжна організація національної системи масових електронних платежів. [Електронний ресурс]. – Режим доступу: http://zakon5.rada.gov.ua/laws/show/va119500-08

28. Hryshchuk R., Yevseiev, S. Shmatko A. Construction methodology of information security system of banking information in automated banking systems: monograph, 284 p., Vienna.: Premier Publishing s. r. o., 2018.

29. Serhii Yevseiev, Alla Havrylova, Olha Korol, Oleh Dmitriiev, Oleksii Nesmiian, Yevhen Yufa. Research of collision properties of the modified UMAC

algorithm on crypto-code constructions. “EUREKA: Physical Sciences and Engineering”, № 1. – 2022. P. 34–44.

30. Д. Домарєв, В. Домарєв та С. Прокопенко, “Методика оцінювання захищеності інформаційних систем за допомогою СУІБ “Матриця”, Захист інформації, том 15, №1, с. 80 – 86, 2013.

31. С. В. Павленко, “Метод оцінки захищеності інформаційних систем”, Системи озброєння і військова техніка, № 4(20), с. 149 – 154, 2009.

32. Корченко О.Г. Системи захисту інформації: Монографія / Корченко О.Г. - К.: НАУ, 2004. – 264 с.

33. Serhii Yevseiev, Stanislav Milevskyi, Leonid Bortnik, Voropay Alexey, Kyrylo Bondarenko, Serhii Pohasii. Socio-Cyber-Physical Systems Security Concept. 4th International Congress on Human-Computer Interaction, Optimization and Robotic Applications. June 9-11, 2022, Ankara, Turkey.

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.056
О. Назарук
(Тернопільський національний технічний університет імені Івана Пулюя)

СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ТА БЕЗПЕКИ WEB-СЕРВЕРІВ

O. Nazaruk

CREATION SOFTWARE OF WEB SERVER SECURITY ANALYSIS

Комп'ютерні системи та телекомунікації забезпечують надійність функціонування великої кількості інформаційних систем різноманітного призначення. Більшість таких систем несуть у собі інформацію конфіденційного характеру. Компрометація і, як наслідок, порушення штатного режиму функціонування всієї системи можуть призвести до значних матеріальних збитків.

Розвиток держави неодмінно пов'язаний з розвитком ринкових відносин та рентабельної конкурентоспроможної економіки, у якій банківський сектор грає головну роль.

Банківський сектор останнє десятиліття із стрімким ростом цифрових технологій, значно збільшив перелік послуг, які надаються державним інститутам, населенню завдяки розвитку Національної системи електронних платежів, функціями якої є переказ коштів згідно операцій, ініційованих використанням платіжних карток, забезпечення високого рівня безпеки, надійності, швидкості та економічної ефективності виконання операцій з використанням платіжних карток.

Банківська система України як незалежної держави, що прагне до створення розвиненої ринкової економіки, є дворівневою. До першого рівня належить Національний банк, що являє собою «банк банків», центральний банк країни, а до другого – система комерційних банків.

Процес створення сучасної банківської системи, яка відповідала б умовам ринкової економіки, в Україні ще триватиме довго бо в українській економіці ще не відбулася необхідна трансформація і вона не стала по-справжньому ринковою.

Проведений аналіз організаційної структури НСМЕП показав, що в основі виконання функцій її роботи лежить автоматизована карткова систем. Система відноситься до складних багаторівневих систем управління критичного застосування, де передача інформації потребує контролю безпеки на кожному рівні.

Дана система інтегрується у банківські системи та безліч типів терміналів, у тому числі переносні, які працюють в автономному режимі, та банкомати, які виконують більш широкий спектр функцій. НСМЕП управляє потоками електронних грошей, зв'язком терміналів та локальних мереж. Для забезпечення надійної роботи електронна платіжна система повинна бути надійно захищена. З точки зору інформаційної безпеки у НСМЕП існують такі вразливі місця:

– пересилання платіжних та інших повідомлень між банком та клієнтом та між банками;

– обробка інформації в межах організацій відправника та одержувача повідомлень;

– доступ клієнтів до засобів, акумульованих на рахунках.

Основою комплексної АБС та НСМЕП є банківська інформація - сукупність відомостей, пов'язаних зі Статутними документами та Керівництвом банківської установи, організаційно-правовою формою банківської установи, нинішнім виглядом банківської установи та її службовців, видами та формами банківського

ДОДАТОК Б

Логотип програми “BAT”

Фрагмент лістингу програми

Алгоритм парсингу ПЗ “BAT”

```
#!/usr/bin/env python3
import argparse
import os
import sys
import subprocess
import time

from src import output
from src import banners
from src import connection
from src import files
from src import scan_command
from src import list_command

#inputfile = './run.sh'
#myfile = open(inputfile, mode='r' encoding='ascii')
#command="sh run.sh"
#a = input("please enter")

#cmd="bash test.py"
#os.system(cmd)
cmd = "bash test.py  &"

def show(result, content, category, url='', status=True):
    if len(result) > 0:
        if hasattr(args, "output"):
```

```
            output.results(result, content, category,
fields=args.print,
                           status=status,
output=args.output, url=url)
        else:
            output.results(result, content, category,
fields=args.print,
                           status=status)


def check(response, url, category):
    if category == "good":
        content = files.read_json("headers/good.json")
    elif category == "bad":
        content = files.read_json("headers/bad.json")

    result = scan_command.check(response, content,
category=category,
                                status=args.status,

headers_to_analyze=args.headers)
    result = scan_command.ignore_headers(result,
args.ignore_headers)

    show(result, content, category, url,
status=args.status)


def list_headers(args):
    bad = files.read_json("headers/bad.json")
    god = files.read_json("headers/good.json")
```

```
    result = dict()

    if args.headers:
        result["bad"] = list_command.list_headers(bad,
headers=args.headers)
        result["good"] = list_command.list_headers(good,
headers=args.headers)
    else:
        result["bad"] = list_command.list_headers(bad)
        result["good"] = list_command.list_headers(good)

    show(result["bad"], bad, "bad")
    show(result["good"], good, "good")


def scan_headers_url(url, args, index=0, urls_qtd=1):
    if "://" not in url:
        url = "http://" + url

    response = connection.get(url,
redirects=args.no_redirect,
                              user_agent=args.user_agent,
                              insecure=args.insecure)

    if index == 0:
        output.print_header(url, args.print, args.output)

    if args.verbose:
        output.verbose(response, args.verbose)

    response = {header.lower(): value.lower() for header,
```

```
                value in response.headers.items()}

    if args.bad:
        check(response, url, category="bad")
    elif args.good:
        check(response, url, category="good")
    elif args.all:
        check(response, url, category="bad")
        check(response, url, category="good")

    if index + 1 == urls_qtd:
        output.print_footer(args.output)


def scan_headers(args):
    if args.no_explanation:
        output.help()

    if os.path.isfile(args.url):
        urls = files.read_lines(args.url)
        urls_qtd = len(urls)
        for i, url in enumerate(urls):
            scan_headers_url(url, args, index=i,
urls_qtd=urls_qtd)
    else:
        scan_headers_url(args.url, args)


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description="BAT -
Protect Your Server ")
```

```
    sub_parsers = parser.add_subparsers(help="sub-command
help")

    list_parser = sub_parsers.add_parser("list",
aliases=['l'],
                                         help="show a list
of available"
                                              " headers in
bat catalog (that"
                                              " can be
used in scan subcommand"
                                              "-H
option)")
    list_parser.add_argument("-p", "--print",
default=False, nargs="+",
                             help="a list of additional
information about the"
                             " headers to print. For now
there are two"
                             " options: description and
refs (you can use"
                             " either or both)")
    list_parser.add_argument("-B", "--no-banner",
action="store_false",
                             default=True, help="don't
print the bat banner")

    group_headers_list =
list_parser.add_mutually_exclusive_group()
```

```
    group_headers_list.add_argument("-a", "--all",
action="store_true",
                                    default=True,
help="list all available "
                                    "headers [default]")
    group_headers_list.add_argument("-H", "--headers",
nargs="+",
                                    help="a list of
headers to look for in"
                                    " the bat catalog")

    list_parser.set_defaults(command=list_headers)

    scan_parser = sub_parsers.add_parser("scan",
aliases=['s'],
                                         help="scan url to
hardening headers")
    scan_parser.add_argument("-v", "--verbose",
action="count", default=0,
                             help="increase output
verbosity: -v print"
                             " response headers, -vv print
response and"
                             " request headers")
    scan_parser.add_argument("-a", "--all",
action="store_true", default=True,
                             help="scan all cataloged
headers [default]")
    scan_parser.add_argument("-g", "--good",
action="store_true",
```

```
                                help="scan good headers
only")
    scan_parser.add_argument("-b", "--bad",
action="store_true",
                                help="scan bad headers only")
    scan_parser.add_argument("-H", "--headers",
default=False, nargs="+",
                                help="scan only these headers
(see available in"
                                " list sub-command)")
    scan_parser.add_argument("-p", "--print",
default=False, nargs="+",
                                help="a list of additional
information about the"
                                " headers to print. For now
there are two"
                                " options: description and
refs (you can use"
                                " either or both)")
    scan_parser.add_argument("-i", "--ignore-headers",
default=False,
                                nargs="+",
                                help="a list of headers to
ignore in the results")
    scan_parser.add_argument("-B", "--no-banner",
action="store_false",
                                default=True, help="don't
print the bat banner")
    scan_parser.add_argument("-E", "--no-explanation",
action="store_false",
                                default=True,
```

```
                                help="don't print the bat
output explanation")
    scan_parser.add_argument("-o", "--output",
                                choices=["normal", "csv",
"json"],
                                default="normal",
                                help="choose which output
format to use "
                                "(available: normal, csv,
json)")

    scan_parser.add_argument("-n", "--no-redirect",
action="store_false",
                                help="don't follow http
redirects")
    scan_parser.add_argument("-u", "--user-agent",
                                help="set user agent to scan
request")
    scan_parser.add_argument("-k", "--insecure",
action="store_false",
                                help="don't verify SSL
certificate as valid")

    groupOutput =
scan_parser.add_mutually_exclusive_group()
    groupOutput.add_argument("-r", "--recommendation",
action="store_true",
                                default=True,
                                help="output only
recommendations [default]")
```

```
    groupOutput.add_argument("-s", "--status",
action="store_true",
                             help="output actual status
(eg: existent"
                             " headers only)")

    scan_parser.add_argument("url", help="url to look
for")
    scan_parser.set_defaults(command=scan_headers)

    args = parser.parse_args()

try:
    os.system(cmd)
    if hasattr(args, 'no_banner') and args.no_banner:
        output.banner(banners.get_banner())

    if isinstance(args.headers, list):
        args.headers = {header.lower() for header in
args.headers}

    if 'command' in args:
        args.command(args)
    else:
        parser.print_usage()
except Exception:
    print('')
```

Класс Connection.py

```
import requests
from requests.packages.urllib3.exceptions import
InsecureRequestWarning


def get(url, redirects=True, user_agent='bat',
insecure=True):
    if not insecure:

requests.packages.urllib3.disable_warnings(InsecureRequest
Warning)

    headers = {'user-agent': user_agent}
    return requests.get(url, allow_redirects=redirects,
headers=headers, verify=insecure)
```

Класс Files.py

```
import json


def read_lines(f):
    with open(f) as content:
        lines = [line.rstrip('\n') for line in content]
        return lines


def read_json(f):
    with open(f) as content:
        data = json.load(content)
```

```
        return data
```

Класс list_command.py

```
def list_headers(catalog, headers=True):
    result = []
    if isinstance(headers, set):
        for header in catalog:
            if header in headers:
                result.append(header)
    else:
        result = catalog.keys()

    return set(result)
```

Класс output.py

```
import json
from urllib.parse import urlparse
from colorama import init, Fore, Style

init()


def help():
    print('''Output explanation:
    {b}{g}[+]{reset} Good headers. Already used in your
website. Good job!
    {b}{y}[+]{reset} Good headers. We recommend applying
it
```

```
    {b}{r}[-]{reset} Bad headers. We recommend remove
it\n'''.format(b=Style.BRIGHT, g=Fore.GREEN,
y=Fore.YELLOW, r=Fore.RED, reset=Style.RESET_ALL))


def banner(b):
    print(Style.BRIGHT, end='')
    print(b)
    print('https://github.com/islamovmagomed/bat')
    print(Style.RESET_ALL)


def print_bright(message, end='\n'):
    print(Style.BRIGHT + message + Style.RESET_ALL,
end=end)


def print_color(message, color):
    if color == 'red':
        color = Fore.RED
    print(color + message + Fore.RESET)


def get_status(category, status):
    if category == 'good' and status:
        return 'applied'
    elif category == 'good' and not status:
        return 'touse'
    elif category == 'bad':
        return 'toremove'
```

```
def print_line(message, level=1, category = None, title =
None, status=False):
    sts = get_status(category, status)

    if sts == 'applied':
        color = Fore.GREEN
        pre = '[+] '
    elif sts == 'touse':
        color = Fore.YELLOW
        pre = '[+] '
    elif sts == 'toremove':
        color = Fore.RED
        pre = '[-] '
    else:
        color = ''
        pre = ''

    if title:
        print(' '*4*level + Style.BRIGHT + title + ': ' +
Style.RESET_ALL + message)
    else:
        print(' '*4*level + color + Style.BRIGHT + pre +
Fore.RESET + message)


def print_header(url, fields, output):
    if output == 'normal':
        print_bright(url)
    elif output == 'csv':
        csv_header(fields)
```

```
    elif output =='json':
        print('[\n\t{\'url\': '' + url + '','')
        print('\t\'headers\': [')


def print_footer(output):
    if output == 'json':
        print('\t]}\n]')


def csv_header(fields, delimiter=';'):
    print('url' + delimiter + 'status' + delimiter +
'title', end='')

    if isinstance(fields, list):
        for field in fields:
            print(delimiter + field, end='')

    print()


def show(content, fields, category, status=False,
output='normal', url=''):
    if output == 'normal':
        show_normal(content, fields, category, status)
    elif output == 'csv':
        show_csv(content, fields, category, status, url)
    elif output == 'json':
        show_json(content, fields, category, status)
```

```
def show_json(content, fields, category, status):
    sts = get_status(category, status)

    result = {
        'title': content['title'],
        'status': sts
    }

    if isinstance(fields, list):
        if 'description' in fields:
            result['description'] = content['description']
        if 'refs' in fields:
            result['refs'] = content['refs']

    print('\t\t' + json.dumps(result) + ',')


def show_csv(content, fields, category, status, url,
delimiter=';'):
    sts = get_status(category, status)

    if sts == 'applied':
        status = 'applied'
    elif sts == 'touse':
        status = 'recommended to use'
    elif sts == 'toremove':
        status = 'recommended to remove'

    print(url, end=delimiter)
    print(status, end=delimiter)
    print(content['title'], end='')
```

```
    if isinstance(fields, list):
        if 'description' in fields:
            print(delimiter + content['description'],
end='')
        if 'refs' in fields:
            print(delimiter, end='')
            for item in content['refs']:
                print(item + ' ', end='')
    print()


def show_normal(content, fields, category, status):
    print_line(content['title'], category=category,
status=status)

    if isinstance(fields, list):
        if 'description' in fields:
            print_line(content['description'], level=2,
title='Description')
        if 'refs' in fields:
            print_line('', level=2, title='Refs')
            for item in content['refs']:
                print_line(Style.RESET_ALL + item,
level=3)
    print(Style.RESET_ALL, end='')


def results(result, catalog, category, fields=0,
status=False, output='normal', url=''):
    for header in result:
```

```
        if header not in catalog:
            print_bright(header, end='')
            print_color(" isn't an option. See available
option to -H using 'list -a'", 'red')
            exit()

        if isinstance(result, dict) and
isinstance(result[header], list):
            for i in result[header]:
                show(catalog[header][i], fields, category,
status=status, output=output, url=url)
        elif isinstance(catalog[header], list):
            for item in catalog[header]:
                show(item, fields, category,
status=status, output=output)
        else:
            show(catalog[header], fields, category,
status=status, output=output, url=url)


def verbose(response, verbose):
    req = response.request
    if verbose >= 2:
        print_line(req.method + ' ' + req.path_url + '
HTTP/1.1')
        print_line('Host: ' +
urlparse(response.url).netloc)
        for header in req.headers:
            print_line(req.headers[header], title=header)
        print()
```

```
    if verbose >= 1:
        print_line('HTTP/1.1 ' + str(response.status_code)
+ ' ' + response.reason)
        for header in response.headers:
            print_line(response.headers[header],
title=header)
        print()
```

Класс scan_command.py

```
import re


def check_condition(header, rule):
    if isinstance(rule['condition'], str) and
re.search(rule['condition'], header):
        return True
    elif isinstance(rule['condition'], bool) and
rule['condition']:
        return True
    else:
        return False


def clear_bad_headers(intersect, headers, catalog):
    result = dict()
    for header in intersect:
        if isinstance(catalog[header], list):
            for index, rule in enumerate(catalog[header]):
                if check_condition(headers[header], rule):
```

```
                    if header in result:
                        result[header].append(index)
                    else:
                        result[header] = [index]
            else:
                if check_condition(headers[header],
catalog[header]):
                    result[header] = True

    return result


def check(headers, catalog, category='good', status=False,
headers_to_analyze=False):
    if isinstance(headers_to_analyze, set):
        lookFor = headers_to_analyze
    else:
        lookFor = catalog.keys()

    if category == 'good':
        if status:
            result = set(lookFor) & set(headers.keys())
        else:
            result = set(lookFor) - set(headers.keys())
    elif category == 'bad':
        result = set(lookFor) & set(headers.keys())
        result = clear_bad_headers(result, headers,
catalog)

    return result
```