ЗМІСТ

[Перелік позначень та скорочень 4](#_Toc470783612)

[Вступ 5](#_Toc470783613)

[1 Дослідження існуючих видів шкідливого ПЗ та методів його виявлення 7](#_Toc470783614)

[1.1 Види шкідливого ПЗ та їх особливості 7](#_Toc470783615)

[1.2 Стратегії зараження файлів 11](#_Toc470783616)

[1.3 Особливості існуючих методів захисту та виявлення шкідливого ПЗ 15](#_Toc470783617)

[1.3.1 Методи сигнатурного виявлення 15](#_Toc470783618)

[1.3.2 Методи проактивного захисту 17](#_Toc470783619)

[1.4 Аналіз сучасних відкритих open-source систем виявлення шкідливого ПО 20](#_Toc470783620)

[1.5 Постановка задачі 22](#_Toc470783621)

[2 Математичні моделі та алгоритмічне забезпечення програмного модуля для виявлення шкідливого коду 24](#_Toc470783622)

[2.1 Характеристики та параметри систем виявлення шкідливого ПО 24](#_Toc470783623)

[2.1.1 Характеристики системи виявлення вторгнень 24](#_Toc470783624)

[2.1.2 Характеристики методів виявлення вторгнень 26](#_Toc470783625)

[2.2 Формальна модель веб-сторінки 28](#_Toc470783626)

[2.3 Нейронні мережі та їх використання у побудові імунних систем 30](#_Toc470783627)

[2.3.1 Принцип функціонування штучного нейрона 30](#_Toc470783628)

[2.3.1 Властивості нейронних мереж 33](#_Toc470783629)

[2.3.2 Модель функціонування компоненту нейронної мережі у програмному модулі 34](#_Toc470783630)

[3 РОЗРОБКА АРХІТЕКТУРИ програмного модулЮ для виявлення шкідливого коду 38](#_Toc470783631)

[3.1 Онтологія проекту 38](#_Toc470783632)

[3.2 Бізнес-логіка проекту 42](#_Toc470783633)

[3.3 Діаграма прецедентів 43](#_Toc470783634)

[3.4 Діаграма IDEF0 44](#_Toc470783635)

[3.5 Проектування архітектури системи 46](#_Toc470783636)

[3.5.1 Діаграма розгортання компонентів 46](#_Toc470783637)

[3.5.2 Вибір архітектурних патернів 48](#_Toc470783638)

[Висновки 50](#_Toc470783639)

[Список джерел інформації 51](#_Toc470783640)

# Перелік позначень та скорочень

ПЗ – Програмне Забезпечення

ОС – Операційна Система

СВВ – Система Виявлення Вторгнень

IPS – Intrusion Prevention System

HIPS – Host-Based Intrusion Prevention System

ЦПП – Центральний Процесорний Пристрій

RAT – Remote Access Tool

FTP – File Transfer Protocol

ISP – Internet Service Provider

ARP – Address Resolution Protocol

HIDS – Host-Based Intrusion Detection System

APIDS – Application Protocol-Based Intrusion Detection System

NIDS – Network-Based Intrusion Detection System

ПрО – Предметна Область

WPFM – Web-Page Formal Model

MCDS – Malicious Code Detection System

MWPDIS – Malicious Web-Page Detection Intelligent System

NN – Neural Network

# Вступ

В сучасному світі разом зі поширенням використання веб-технологій у повсякденному житті, зростає і кількість шкідливих програм, що розповсюджуються по комп’ютерним мережам. Інтернет-браузери та їх розширення, як і інші додатки, мають певні вразливості, що можуть бути експлуатовані зловмисниками. Основну небезпеку представляють ті, що дозволяють виконувати довільний та підозрілий код у системі жертви.

Основним засобом боротьби зі шкідливими програмами сьогодні є сигнатурний аналіз, тобто виявлення послідовності команд, що погрожують безпеці даних. Конструювання баз даних сигнатур вірусів є предметом діяльності великої кількості організацій, що займаються впровадженням відповідного ПЗ. В основі цього підходу лежить фільтрація шкідливого програмного коду з достатньо високою ефективністю. Проте, у зв’язку з різким ростом об’єму даних, який підлягає фільтрації, цей підхід породжує деякі складності, що пов’язані з необхідністю аналізу всіх оброблених комп’ютером даних. Авжеж, це негативно впливає на продуктивність роботи системи. Також зростає частота появи нових вірусних скриптів, тож з’явилась необхідність більш частого оновлення баз даних вірусних сигнатур. Ці проблеми призводять до необхідності впровадження альтернативних методів виявлення та захисту від шкідливих скриптів.

Одним з напрямків боротьби з вірусами є розробка так званих імунних систем. Імунні системи основані на підходах, що є аналогічними до імунної системи живих організмів. Цей підхід дозволяє комп’ютерним системам «довчитися» в процесі функціонування та надалі самостійно виявляти шкідливий код.

Сьогодні вже існує декілька систем захисту інформації, що працюють по аналогії з імунними системами. Проте, ці системи мають спеціалізований характер, коли задача розробки універсальної системи залишається відкритою, та не позбавлені достатньої кількості погрішностей. Також не можна не зауважити, що і розробка спеціалізованих систем має високу цінність, бо існують окремі класи однотипних вірусних скриптів, захист від яких залишається актуальною задачею.

# 1 Дослідження існуючих видів шкідливого ПЗ та методів його виявлення

У даному розділі буде розглянуто теоретичні основи для аналізу шкідливого ПО, включаючи механізми шкідливих програм для ускладнення виявлення антивірусними програмами, а також їх вплив на інструменти аналізу.

## 1.1 Види шкідливого ПЗ та їх особливості

Шкідливе П3 – це будь-яке ПЗ, що викликає шкоду для користувача, комп'ютера або мережі. [8]

Для виявлення шкідливого коду необхідно володіти даними про поведінку шкідливого ПЗ в залежності від його типу, способу впровадження і зараження системи, і до яких наслідків це призводить. Аналітики шкідливого ПЗ і антивірусні компанії поділяють шкідливі програми за іменами та особливостями поведінки [9]. Розглянуті нижче поведінкові типи не є взаємовиключними: шкідливі програми можуть об'єднати в собі риси кількох

типів.

*Hijacker* змінює налаштування браузера без згоди користувача. Віруси цього типу замінюють стандартну сторінку пошуку (пошукову систему), домашню сторінку, сторінки помилок і додають нові закладки зі своїм контентом. Вони також ускладнюють скасування змін і можуть викликати проблеми, якщо користувач намагається видалити їх.

Приклади: Conduit Search Protect, Babylon Toolbar.

*Trojan* (троянський кінь, троян) – шкідлива програма, яка обманює користувачів для свого запуску, надаючи корисну функціональність або повідомляючи користувачеві, що це корисна і безпечна програма. [10, c.37] Троянські коні комбінуються з додатковими модулями, наприклад, з завантажувачем (downloader), монтажником шкідливого ПО (dropper), запасним входом (backdoor), збирачем інформації (stealer). Троян є найпопулярнішим типом серед шкідливих програм. У квітні 2014 року PandaLabs зареєструвала «6 мільйонів нових шкідливих штамів», з них 78.92% – це трояни. [11]

*Downloader* (завантажувач) є частиною програмного забезпечення, яке завантажує інший шкідливий контент, наприклад, з веб-сайту, встановлює або запускає його. [10, c.39] [8, c.3]

*Dropper* (дроппер) це програма, яка розгортає інше шкідливе ПЗ в файлову систему. Dropper може виконувати процедури по установки та запуску шкідливих програм. [8, с.39-40]. Різниця порівняно з завантажувачем (Downloader) в тому, що Dropper вже містить шкідливий код в собі. [9]

*Rootkit* (руткіт) – програмне забезпечення, мета якого - приховування присутності інших шкідливих програм або дій. [8, с.4] Руткіт може приховувати активність, файли і процеси. Руткіти часто формують в поєднанні з функціональністю бекдорів.

*Backdoor* (запасний вхід, «чорний хід», бекдор) дозволяє отримати доступ до системи в обхід звичайних механізмів захисту доступу. [8, с.3] Бекдор використовується для отримання доступу до системи в будь-який час. Особливим видом бекдору є інструменти віддаленого адміністрування або RAT. [12, с.13] Вони дозволяють отримувати доступ до комп'ютера віддалено і використовуються для шкідливих і не зловмисних цілей. RAT дозволяють службі технічної підтримки віддалено усувати проблеми. Але RAT може використовуватися і зловмисно. Користувачі RAT можуть шпигувати за людьми через веб-камеру і намагатися налякати своїх жертв, контролюючи їх комп'ютер.

*Spammer* – шкідлива програма для розсилки спаму. Cпамери використовують комп’ютер жертви, щоб відправити небажані повідомлення – так званий спам. [8, с.4] [10, с.40-41]). Спамери можуть відправляти свої повідомлення, використовуючи електронну пошту, SMS або пости і коментарі в інтернет-спільнотах.

*Stealer* (інформаційний злодій, збирач) – це шкідлива програма, яка зчитує конфіденційні дані з комп'ютера жертви і відправляє їх зловмисникові. Прикладами інформаційних збирачів є: клавіатурні шпигуни (keylogger), перехоплювачі (sniffer), збирачі хешів паролів, а також деякі види троянів. [8, с.4] Троян-викрадач переконує користувачів, що він доброякісна програма, для того, щоб споживач ввів конфіденційні дані. Прикладом є програма, яка стверджує, що може додати грошей на рахунок PayPal користувача, але, насправді, вона посилає облікові дані PayPal користувача на електронну адресу пошти зловмисника.

*Botnet* – це група різних машин зі встановленим backdoor, які отримують і виконують інструкції з одного сервера. [8, с.3]. Ботнет встановлюється без згоди власників комп'ютерів, і може бути використаний для виконання атак розподіленої відмови в обслуговуванні (DDoS) або для розсилки спаму.



Рисунок 1.1 – Scareware, яке маскується як антивірусне програмне забезпечення

*Scareware* намагається обдурити користувачів, схиляючи до здійснення деякої покупки або лякаючи користувача. [8, с.4]). Типовим прикладом Scareware є програма, яка виглядає як антивірусний сканер і показує користувачеві підроблені попередження про шкідливі програми, які були знайдені у системі. Далі пропонується користувачеві купити дане програмне забезпечення для того, щоб видалити шкідливі програми. Приклад такої програми показаний на Рис. 1.1. Часним випадком Scareware у веб є так зване Adware – шкідливі скрипти на сторінках, що містять нав’язливу рекламу або також фальшиві повідомлення, схиляючи користувачів до здійснення потенціально небезпечних дій.

*Malware construction kit* (конструктор шкідливих програм) – це ПЗ для генерації шкідливої ​​програми або коду шкідливої програми, ґрунтуючись на налаштуваннях користувача. Конструктори шкідливих програм дозволяють людям без знань в програмуванні створювати свої шкідливі програми. Прості конструктори дозволяють змінити тільки деякі налаштування, наприклад, адреси електронної пошти або FTP-аккаунт, куди буде відправлятися знайдена інформація з комп'ютерів жертв. Більш складні комплекти використовують методи контр-аналізу і контр-виявлення, і можуть генерувати широкий спектр різних виконуваних файлів шкідливих програм. Рис. 1.2 показує приклад конструктора шкідливих програм, який має функції контр-аналізу.



Рисунок 1.2 – Приклад Malware Construction Kit

*Virus* розмножується рекурсивно, заражаючи, замінюючи інші програми або редагуючи посилання на ці програми, замінюючи його на код вірусу. [10, с.27, 36] Зазвичай вірус виконується, якщо користувач запустить заражений файл. Такий заражений файл називається хост-файл (з посиланням на термінологію яка використовується для біологічних паразитів). Віруси традиційно поширюються на інші комп'ютери, за допомогою переносних засобів зберігання інформації таких, як, наприклад, USB флеш-накопичувачі, CD або DVD диски. Вірус називається «Germ» якщо він знаходиться в його первісному вигляді, до будь-якої інфекції. [10, с.39] Початкова установка коду Germ робиться за допомогою дроппера, після чого вірус може «повторити на собі» зараження. [10, с. 40]

Так звані «сплячі віруси» в пасивному стані знаходяться на машині, не заражаючи нічого: вони очікують тригера (повідомлення) або відповідної системи для поширення. [12, с.14] Протилежні сплячим вірусам - активні.

*Worm* (хробак) – це мережевий вірус, який для поширення в першу чергу використовує мережі. Як правило, хробаку не потрібен хост-файл, і він виконується сам, без необхідності втручання користувача. [10, с.36] Але є винятки – це хробаки, які поширюються за допомогою пошти, та їм необхідна взаємодія з користувачем. Черв'як є підкласом Вірусу. Згідно з доповіддю PandaLabs про новостворювані шкідливі штами, у квітні 2014 року серед них 7.44% відсотків займали віруси та 10.78% – хробаки. [11]

## 1.2 Стратегії зараження файлів

Зараження будь-якого файлу може привести до мальформації, тому що більшість типів заражень вимагають модифікації хост-файлом. У зв'язку з цим, систематизація знань про стратегії зараження файлів вірусами є дуже важливою, так як для побудови системи виявлення шкідливого ПЗ необхідно володіти інформацією про природу виникнення певних аномалій. Залежно від стратегії зараження іноді можливо видалити або відключити шкідливий код в зараженому файлі. Цей процес називається дезінфекція або лікування і виконується антивірусним програмним забезпеченням. Слід зазначити, що лікування не обов'язково відновляє файл до його оригінальному стану.

Розглянути нижче стратегії зараження працюють на більшості форматів виконуваних файлів, у тому числі і у веб-додатках.



Рисунок 1.3 – Стратегії зараження файлів

*Перезапис* (Overwriting) – найпростіша стратегія вірусів. Вони шукають файли на диску і копіюють своє тіло, перезаписуючи оригінальний файл. [10, с.115] Зараження методом перезапису може завдати серйозної шкоди для системи, тому що відбувається знищення файлів, поверх яких вони записуються, і в даному випадку лікування неможливо. Проте, користувач дуже швидко розпізнає побічні ефекти ушкодження, тому це не дуже успішна стратегія зараження.

Перезаписуючий вірус тільки замінює початок файлу розміром даних, рівних розміру тіла самого вірусу. [10, с.116] Тож заражений файл зберігає розмір оригінального файлу. Перезаписана частину файлу у той час знищується.

*Приєднання в кінці* (Appending) – дана стратегія зводиться до приєднання тіла вірусу в кінець файлу (див. Рис. 1.3) і зміну інструкції переходу (jump) на початку файлу до початку тіла вірусу в кінці файлу. [10, с.117] Зазвичай вірус змінює інструкції переходу після того, як запише копію вірусу в кінець файлу, і таким чином користувач не стає свідком зараження. [10, с.118]

Деякі формати файлів визначають точку входу, яка є адресою точки старту виконуваного коду. Віруси, що приєднуються до цих файлів, змінюють адресу точки входу виконуваного коду на адресу початку вірусного коду, або додають команду переходу відразу після точки старту на початок вірусного коду.

Зазвичай файли після зараження за допомогою цієї стратегії піддаються лікуванню.

*Додавання до початку* (Prepending) – у цьому випадку зараження відбувається прикріпленням тіла вірусу до початку файлу без пошкоджень останнього. [10, с.118-120] Тіло оригінального файлу залишається неушкодженим, тому лікування є можливим.

Щоб приховати зараження від користувача - вірус запускає оригінальний файл, наприклад, копіює його як тимчасовий файл на диск і використовує звернення до функції виклику *system()* для запуску оригінального файлу. [10, с.120]

Підтипом даного зараження є класичне паразитне зараження (див. Рис. 1.3). Вірус, який використовує цю стратегію, замінює початок файлу своїм тілом, а початок, який замінює, приєднує в кінець файлу. [10, с. 120-121].

*Додавання в «печеру»* (Cavity Infection) – відповідно до даної стратегії, вірус записує своє тіло найчастіше без перезапису «корисного» коду оригінального файлу. [10, с. 121] Відбувається перезапис тільки тих частин, містять нулі - в так званих «печерах» (caves) файлу [10, с.121]. Наприклад, «печера» може бути між заголовком і фактичним початком коду файлу. Якщо вірус інфікує цю область - це називається зараженням заголовка (див. «Cavity A» на Рис. 1.3). Віруси, які заражають єдину «печеру», повинні бути досить маленькими, що б відповідати розміру «печери» і не вийти за її область.

Такий вірус зазвичай зберігає оригінальну точку входу оригінального файлу і ставить елемент управління, що вказує на свій код після основного коду.

Підтипом даного зараження є фракціоноване зараження (див. «Cavity B» на Рис. 1.3), де вірус розбиває себе на частини і вписує фрагменти свого коду в кілька «печер». Перша частина містить код завантажувача, який є відповідальним за збірку частин в єдиний вірус в пам'яті під час виконання програми [10, с.122-123].

Відновлення вихідного файлу після зараження може бути складним, а іноді і не піддається відновленню, якщо перезаписані частини неможливо відновити.

*Амебне зараження* (Amoeba Infection) – ця стратегія зараження використовується рідко. [10, с.124] Вірус розділяє себе на 2 частини. «Голова» приєднується до початку файлу, а друга частина приєднується до краю. Якщо файл запускається - «голова» запускає другу частину в кінці файлу.

У деяких випадках вірус створює тимчасову копію оригінального файлу і запускає його в випадку запуску зараженого файлу.

*Зараження компаньйона* (Companion Infection) – у випадку даної стратегії вірус не змінює оригінальний файл. Він використовує порядок запуску виконуваних файлів операційної системи для свого запуску замість оригінального файлу.

Існує 3 види вірусів, що використовують цю стратегію:

1. «Регулярний компаньйон» розміщується в тому ж каталозі, що і основний файл, маючи ідентичне ім'я, але з іншим розширенням файлу. Якщо користувач вказує для запуску тільки ім'я файлу без розширення - «регулярний компаньйон» виконається замість основного файлу. Наприклад, файли з розширенням .COM шукаються перед файлами з розширенням .EXE на MS-DOS. Вірус «Компаньйон» може заразити файли .EXE, розміщуючи себе в тому ж каталозі і з тим же ім'ям файлу з розширенням .COM. [13, секція 2.2.1]

2. «Компаньон PATH (шлях)» користується оглядом каталогів, який використовується операційною системою для пошуку файлів. Операційна система зазвичай використовує змінну PATH при пошуку для визначення пошукових запитів виконуваних файлів. «Компаньйон PATH» має те ж ім'я, що і оригінальний файл, і розміщує себе в каталозі, який знаходиться «вище», ніж каталог оригінального файлу. [13, секція 2.2.2] Наприклад, віруси наслідують відкритим текам DLL. Додатки, які імпортують функції з DLL, можуть завантажити функціонал вірусу замість функціоналу справжнього DLL файлу.

3. «Поєднуючий компаньйон» користується макросами командного рядка, які визначає користувач, вони ж псевдоніми (aliases). [13, секція 2.2.2] Псевдоніми використовуються як ярлики для довгих командних послідовностей. Вірус може створити псевдонім, який замінить загальну команду для запуску «Компаньйона». [13, секція 2.2.3].

Вірус «компаньйон» існує поруч з оригінальним файлом. Видалення файлу вірусу або псевдоніму в разі «поєднуючого компаньйона» зупинить зараження.

## 1.3 Особливості існуючих методів захисту та виявлення шкідливого ПЗ

Шкідливе ПЗ призначене для несанкціонованого вторгнення до обчислювальних ресурсів комп’ютерної системи або мережі з метою їх несанкціонованого використання та/або вчинення шкоди їх власнику за допомогою видалення, копіювання, підміни інформації. [2]

Як основні методи захисту від впливу шкідливого коду буде розглянуто такі методи, як сигнатурний аналіз і проактивний захист.

### 1.3.1 Методи сигнатурного виявлення

Сигнатура атаки (вірусу) – це характерні ознаки атаки або вірусу, що використовуються для їх виявлення. Більшість сучасних антивірусів, сканерів вразливостей та СВВ використовують «синтаксичні» сигнатури, взяті безпосередньо з тіла атаки (файла вірусу або мережевого пакета, що належить експлойту). Також існують сигнатури, засновані на поведінці або аномаліях - наприклад, занадто агресивне звернення до якого-небудь мережевого порту на комп'ютері. [3]

Сигнатури створюються в результаті кропіткого аналізу декількох копій файлу, що належать одному вірусу. Сигнатура повинна містити тільки унікальні рядки з цього файлу, що гарантують мінімальну можливість помилкового спрацьовування.

Виявлення, основане на сигнатурах, є основою функціонування переважної більшості антивірусних програм і СВВ. За цим методом програма проводить аналіз файлу, пакета чи фрагмента, звертається до так званого «словника» існуючих вірусів. Якщо виявлено відповідність до сигнатури шкідливого коду зі словника, антивірусне ПЗ може ініціювати виконання одної з наступних операцій:

– видалення інфікованого файлу;

– відправлення файлу до «карантину», що має на увазі зробити недоступним для виконання з метою недопущення розповсюдження вірусу;

– відновлення файлу, що має на увазі видалення вірусного фрагменту з тіла файлу.

Антивірусне ПЗ, розроблене на основі метода виявлення відповідності до сигнатури, як правило, обов’язково проводить аналіз файлу у тому випадку, якщо система створює, відкриває або відправляє файл через мережу Інтернет. Таким чином, віруси можуть бути виявлені одразу після занесення до комп’ютера та до того, як вони вчинять певної шкоди.

Розробка сигнатур важко піддається автоматизації. Незважаючи на масу досліджень, присвячених автоматичної генерації сигнатур, наростаючий поліморфізм вірусів і атак лишають синтаксичні сигнатури сенсу. Існують так звані «метаморфічні» віруси, в яких деякі фрагменти коду модифікуються або шифруються так, щоб неможливо було виявити збіг з визначенням в словнику вірусів.

Таким чином, основною перевагою методу сигнатурного аналізу є можливість визначення конкретної атаки з високою точністю та малою ймовірністю помилкового виклику. Проте, цей метод має і наступних ряд недоліків:

– неспроможність виявлення деякі нові атаки;

– неспроможність захисту від поліморфних та метаморфічних вірусів;

– необхідність регулярного та оперативного оновлення;

– необхідність ручного аналізу вірусів, відсутність автоматизованого підходу.

### 1.3.2 Методи проактивного захисту

В даний час проактивні технології є важливим і невід'ємним компонентом антивірусного програмного забезпечення. Більш того, як правило, в антивірусних продуктах використовується поєднання відразу декількох технологій проактивного захисту. Наприклад, евристичний аналіз і емуляція коду успішно поєднуються з поведінковим аналізом, що дозволяє багаторазово підвищити ефективність сучасного антивірусного ПЗ проти нових шкідливих програм. Основною метою проактивних методів, на відміну від методу сигнатурного аналізу, є запобігання інфікування системи, а не пошук вже існуючого у системі вірусу.

Перспективним напрямком є дослідження проактивних технологій захисту телекомунікаційних систем і мереж від шкідливого П3 та розробка СВВ та їх основі.

Як основні методи проактивного захисту розглянемо евристичний аналіз та технології емуляції, поведінкового аналізу, обмеження привілеїв, віртуалізації робочого оточення, забезпечення безпеки на основі політик ті ін. [4 - 6]

*Технологія евристичного аналізу* дозволяє на основі аналізу коду, скрипта або макросу виявити фрагменти коду, що відповідають за шкідливу активність. Евристичним аналізатором називається набір підпрограм, які аналізують код виконуваних файлів, макросів, скриптів, пам'яті або загрузочних секторів для виявлення в ньому різних типів комп'ютерних програм, не обумовлених звичайними (сигнатурними) методами. Евристичні аналізатори призначені для пошуку невідомого шкідливого ПЗ.

Ефективність даної технології є низькою, що обумовлено великою кількістю помилкових спрацьовувань, а також занадто великим набором технік, що використовуються авторами шкідливого програмного забезпечення для задля функціонування евристичного компонента антивірусного програмного забезпечення.

*Технологія емуляції* дозволяє запускати додаток в середовищі емуляції, що емулює поведінку операційної системи або ЦПП. При виконанні програми в режимі емуляції не має можливості завдання шкоди системі користувача, бо шкідливий вплив буде детектовано емулятором. Незважаючи на гадану ефективність даного підходу, він також не позбавлений недоліків:

– емуляція займає занадто багато часу і ресурсів, що негативно позначається на швидкодії виконання повсякденних операцій;

– сучасні шкідливі програми здатні виявляти імітоване середовище і припиняти своє виконання в ній.

*Технологія аналізу поведінки* ґрунтується на перехопленні всіх важливих системних функцій або установці так званих міні-фільтрів, що дозволяють відстежувати всю активність у системі користувача. Поведінковий аналіз дозволяє оцінювати не тільки одиничну дію, але і послідовність дій, що багаторазово підвищує ефективність протидії вірусним загрозам. Ця технологія є основою для цілого класу програм – поведінкових блокіраторів (HIPS).

*Технологія обмеження привілеїв* виконання працює за принципом обмеження активності потенційно шкідливих додатків таким чином, щоб вони не могли завдати шкоди системі користувача. Обмеження активності досягається за рахунку виконання невідомого П3 в обмеженому середовищі, звідки додаток не має прав доступу до критичних системних файлів, реєстру та іншої важливої ​​інформації. Ця технологія є ефективною у протидії сучасним загрозам, проте має значний недолік: користувач повинен володіти знаннями, необхідними для правильної оцінки невідомого додатку.

*Технологія віртуалізації робочого оточення* працює за допомогою системного драйвера, який перехоплює всі запити на запис на жорсткий диск і замість цієї операції виконує запис в спеціальну дискову область - буфер. Таким чином, навіть в тому випадку, якщо користувач запустить шкідливе ПЗ, воно проживе не далі ніж до очищення буфера, який як правило виконується при вимкненні комп’ютера. Проте слід зазначити, що цей підхід не може захистити від шкідливого ПЗ, що має мету здобути конфіденціальну інформацію, бо доступ до читання з жорсткого диску не забороняється.

*Технологія забезпечення безпеки на основі політик*. Політика безпеки є необхідним атрибутом будь-якої продуманої стратегії захисту від ІТ-загроз. Продумана політика дозволяє в кілька разів зменшити ризик зараження шкідливою програмою, атаки хакерів або витоку конфіденційної інформації. Проте, на формування політики таож потрібен час (не завжди менше, ніж на аналіз вірусу для додавання процедур виявлення в антивірусну базу), що тягне за собою приступність проміжку часу, в який комп’ютер користувача залишається незахищеним. Сьогодні існує декілька підходів до такого способу забезпечення безпеки, які розглянуто нижче.

– Підхід Trend Micro: Trend Micro Outbreak Prevention Services.

Даний сервіс передбачає поширення деяких політик, що дозволяють запобігти епідемії. Тобто політика розповсюджується до появи оновлення антивірусних баз або патчів. Недолік є в тому, що швидкість додавання процедур виявлення нового шкідливого коду в Trend Micro доволі низька.

– Підхід Cisco-Microsoft.

У основі цбого підходу лежить обмеження доступу в корпоративну мережу для комп'ютерів, які не відповідають політиці безпеки компанії (наприклад, відсутні необхідні оновлення ОС, немає останніх оновлень антивірусних баз і т.д.). Для приведення комп'ютера до відповідності політиці виділяється доступ тільки на спеціальний сервер оновлень. Після установки всіх необхідних оновлень і виконання інших дій, необхідних політикою безпеки, комп'ютер отримує доступ в корпоративну мережу.

– Технологія забезпечення безпеки на основі IPS.

Системи запобігання вторгнень (IPS) передбачають можливість закриття найбільш часто використовуваних шкідливими програмами вражливостей комп'ютера перед новою загрозою ще до виходу оновлення антивірусних баз: блокування портів, тобто можливості попадання інфекції на комп'ютер і її подальшого розмноження; створення політик для обмеження доступу до директорій або окремих файлів; виявлення джерела інфекції в мережі і блокування подальших комунікацій з ним. Дана технологія розрахована на роботу проти атак хакерів і безфайлових вірусів, але проти класичних і поштових вірусів і троянських програм IPS не є ефективною.

*Технологія забезпечення безпеки на основі захисту від переповнення буфера.* Ідея технології - не допустити переповненнябуфера для найбільш поширених програм,сервісів Windows, включаючи Word, Excel, InternetExplorer, Outlook і SQL Server. Більшість сучасних атак направлено на вразливості, що пов’язані з переповненням буфера.

*Технологія забезпечення безпеки на основі поведінкових блокувань.* Основна ідея блокіратора – аналіз поведінки програм і блокування виконання будь-яких небезпечних дій. Теоретично блокіратор може запобігти поширенню будь-якого, як відомого, так і невідомого вірусу. Саме в цьому напрямку і рухається більшість розробників антивірусного ПЗ. Останнім часом більшість систем запобігання поширенню поштових вірусів за механізмом є поведінковими блокіраторами.

Перше покоління поведінкових блокіраторів з'явилося ще в середині 90-х років. При виявленні небезпечної активності ставилося питання користувачу – дозволити або заборонити дію. При багатьох випадках такий підхід працював, але підозрілі дії виробляли і легітимні програми (аж до операційної системи) і якщо користувач не мав належної кваліфікації, питання антивіруса викликали нерозуміння. З проникненням персональних комп'ютерів все глибше в повсякденне життя знижувався середній рівень кваліфікації користувачів, і перші поведінкові блокатори перестали бути потрібні. Друге покоління поведінкових блокіраторів відрізняє те, що вони аналізують не окремі дії, а послідовність дій, і вже на підставі цього робиться висновок про шкодочинність ПЗ. Це значно скорочує кількість запитів до користувача і підвищує надійність детектування.

## 1.4 Аналіз сучасних відкритих open-source систем виявлення шкідливого ПО

У даному розділі розглянуті доступні на сьогоднішній день системи виявлення атак з відкритим вихідним кодом.

Всього розглянуто 4 системи виявлення атак. У Табл. 1 наведена коротка інформація по кожній з них.

Частина розглянутих систем (Bro) розроблені в університетах і базуються на дослідженнях в області виявлення атак, проведених в цих університетах.

Таблиця 1 – Відкриті системи виявлення атак

|  |  |  |
| --- | --- | --- |
| Назва системи | Виробник | Посилання |
| Bro | University of California, Lawrence Berkeley National Laboratory | https://www.bro.org/ |
| OSSEC | Daniel B. Sid , OSSEC.net | http://ossec.github.io/ |
| Prelude | Yoann Vandoorselaere  Laurent Oudot | https://www.prelude-siem.org/ |
| Snort | Martin Roesch | https://www.snort.org/ |

Всі розглянуті системи використовують в якості основного методу виявлення атак сигнатурний метод (порівняння рядків, шаблонів).

Система Bro використовує регулярні вирази над трасами, які формуються мережевими протоколами. Набір регулярних виразів створюється експертами. Крім того, до складу системи входить транслятор сигнатур з формату системи Snort в сценарії Bro (хоча в даний час цей транслятор підтримуєnне всі конструкції мови Snort). [15]

Система OSSEC є монолітною - в сенсори і аналізатори «зашиті» знання розробників системи виявлення атак про те, які послідовності повідомлень в журналах можуть бути ознаками атаки. Така архітектура системи є важко розширюємою з точки зору бази знань про атаки. [16]

Система Prelude використовує різні аналізуючі компоненти для мережевих даних і журналів реєстрації. Для аналізу мережевих даних може використовувати систему Snort. Також використовується набір спеціалізованих модулів для виявлення специфічних атак, таких як сканування портів, некоректних ARP-пакети і т.п. Спеціальні модулі виробляють дефрагментацію IP, складання TCP-потоку, декодування HTTP-запитів. [17]

Система Snort використовує базу сигнатур відомих атак. У ній також використовується набір спеціалізованих модулів для виявлення специфічних атак, таких як сканування портів або відправка великої кількості фрагментованих пакетів. Спеціальні модулі виробляють дефрагментацію IP, декодування HTTP-запитів.

Сторонні розробники часто реалізують інші методи виявлення атак у вигляді модулів (препроцесорів) Snort. Але в основну версію системи вони не входять. [18]

## 1.5 Постановка задачі

Об’єктом дослідження даної курсової роботи є шкідливе ПО у веб-середовищі.

Предметом дослідження курсової роботи є методи виявлення шкідливого програмного забезпечення.

У зв’язку з різким ростом об’єму коду, який підлягає фільтрації антивірусним ПО, існуючі підходи до детектування шкідливого коду породжують проблему, що є пов’язаною з необхідністю аналізу всіх оброблених комп’ютером даних. Це негативно впливає на продуктивність роботи системи. Зі зростом частоти появи нових вірусних скриптів, тож з’явилась необхідність більш частого оновлення баз даних вірусних сигнатур. І зараз у функціонуванні антивірусного ПО має місце висока частота хибних спрацьовувань. Ці проблеми призводять до необхідності впровадження альтернативних методів виявлення та захисту від шкідливих скриптів. Тож метою курсової роботи є розробка алгоритмічного забезпечення та архітектурної моделі програмного модулю для детектування шкідливого коду у змісті скриптів веб-сторінок або веб-додатків з використанням технологій машинного навчання.

Для досягнення мети необхідно вирішити заступні задачі:

* проаналізувати предметну область індивідуальної задачі;
* дослідити існуючі методи детектування шкідливого ПО;
* дослідити існуючи види шкідливого ПО як програмні артефакти;
* розробити модель інтернет-сторінки на основі формальних мов, що відрізняється від відомих аналогів тим, що вона забезпечує математичний опис інтернет-сторінки як вектора в n-вимірному просторі ознак;
* дослідити та проаналізувати алгоритми виявлення шкідливих інтернет-сторінок на основі технологій машинного навчання, що відрізняються від відомих аналогів тим, що вони спрямовані на вирішення більш складної задачі класифікації, і в них реалізовані раніше не аналізовані (в контексті поставленого завдання) технології машинного навчання, такі як нейронні мережі;
* провести дослідження моделі нейронної мережі для вирішення задачі виявлення шкідливих інтернет-сторінок.
* розробити первинну онтологію та бізнес-логіку проекту;
* проаналізувати та призвести приблизний набір технологій, що будуть використатися для реалізації проекту;
* побудувати UML-діаграми, що відображують майбутню архітектуру програмного модуля.

# 2 Математичні моделі та алгоритмічне забезпечення програмного модуля для виявлення шкідливого коду

## 2.1 Характеристики та параметри систем виявлення шкідливого ПО

### 2.1.1 Характеристики системи виявлення вторгнень

*Клас атак, що підлягають виявленню.* Цей критерій визначає, які класи атак здатна виявити розглянута система. Це одна з ключових характеристик. У зв'язку з тим, що на сьогоднішній день жодна система не здатна виявити атаки абсолютно всіх класів, для більш повного покриття всього спектра атак необхідно комбінувати різні СВВ. Тут використовується класифікація атак, заснована на поділі ресурсів захищаємої системи за типами.

Клас атаки визначається наступною множиною ознак:

**<L, R, A, D> ,** (1)

де L - розташування атакуючого об'єкта, R – ресурс, що атакується, A - цільовий вплив на ресурс, D – ознака розподіленого характеру атаки. [19] Більш детально ці ознаки розглянуті нижче.

**L = {li****le}.** Визначає розташування атакуючого об'єкта. Може бути або внутрішнім по відношенню до захищає мого об’єкта (**li** – location internal), або зовнішнім (**le** – location external).

**R = {rl rn, ru, rs, rc, rp}.** Визначає ресурс, що атакується. Ресурси поділяються по розташуванню і по типу. За розташуванням: вузлові (**rl** – resource link), мережеві (**rn** – resource net). За типом: призначені для користувача ресурси (**ru** – resource for user), системні ресурси (**rs** – resource for system), ресурси СУБД (**rd** – resource for database), обчислювальні ресурси (**rc** – resource for calculations), ресурси захисту (**rp** – resource for protection).

**A = {as, au, ar, ai, ad}**: Визначає цільовий вплив на ресурс: збір інформації (**as** – affection on sources), отримання прав користувача ресурсу (**au** – affection on user), отримання прав адміністратора ресурсу (**ar** – affection on rights), порушення цілісності ресурсу (**ai** – affection on integrity), порушення працездатності ресурсу (**ad –** affection on disablement).

**D = {dd, dn}**. Ознака розподіленого характеру атаки: розподілена (**dd** - distributed), нерозподіленa (**dn** – non-distributed).

Розглядаючи класи вторгнень, які розроблений у майбутньому програмний модуль буде мати можливість виявляти, маємо приблизно наступну модель:

**< { li }, { rl**  **rn**  **ru }, { as**  **au**  **ai**  **ad }, { dn } > .** (2)

*Рівень спостереження за системою.* Ця характеристика визначає рівень абстракції аналізованих подій в системі, що захищається і визначає межі застосування методу для виявлення атак в мережах. Існують наступні рівні спостереження:

– HIDS: спостереження на рівні операційної системи окремого вузла мережі;

– NIDS: спостереження на рівні мережного взаємодії об'єктів на вузлах

мережі;

– APIDS: спостереження на рівні окремих додатків вузла мережі;

– Hybrid: комбінація спостерігачів різних рівнів.

Далі у рамках даного дослідження буде розглядатися рівень APIDS, який має бути рівнем спостереження за системою розробленого у майбутньому програмного модулю.

*Адаптивність до невідомих атак.* Визначає здатність використовуваного методу виявляти раніше невідомі атаки.

*Масштабованість.* Визначає можливість додавання нових аналізованих ресурсів мережі, нових вузлів и каналів передачі даних, в тому числі можливість управління єдиною розподіленої системою виявлення атак. Управління може бути централізоване та / або розподілене. Додатково може бути присутньою можливість віддаленого управління СВВ. Сюди належать завдання установки, налаштування та адміністрування системи. При повністю розподіленому управлінні необхідно управляти всіма компонентами СВВ окремо. При повністю централізованому управлінні всі компоненти СВВ можуть керувати з одного вузла. Оптимальною є організація управління з централізованого схемою, у якої може бути кілька центрів, и вони можуть динамічно змінюватися.

*Відкритість.* Визначає, наскільки система є відкритою для інтеграції в неї методів виявлення атак, компонентів сторонніх розробників та сполучення її з іншими системами захисту інформації. Це можуть бути програмні інтерфейси для вбудовування додаткових модулів і / або реалізація стандартів взаємодії мережевих компонентів.

*Формування відповідної реакції на атаку.* Перевіряє наявність вбудованих механізмів відповідної реакції на атаку, окрім самого факту її реєстрації. Прикладами реакції можуть бути втрата з'єднання з атакуючім об'єктом, блокування його на міжмережевому екрані, відстеження шляху вторгнення атакуючого об'єкта в систему та ін.

*Захищеність.* Визначає ступінь захищеності СВВ від вторгнень у її компоненти, включаючи захист інформації, що передається, стійкість до часткового виходу компонентів з ладу або їх компрометації. Враховуються такі питання, як наявність вражливостей у компонентах СВВ, захищеність каналів передачі даних між ними, а також авторизація компонентів всередині СВВ.

### 2.1.2 Характеристики методів виявлення вторгнень

*Рівень спостереження за системою.* Інформація про дану характеристику подана у розділі 2.1.1. Свій конкретний рівень спостереження за системою може мати як метод виявлення вторгнень, так і ціла система.

*Верифікованість методу.* Даний критерій дозволяє оцінити, чи може людина (наприклад, кваліфікований оператор СВВ або експерт) відтворити послідовність кроків щодо прийняття рішення про наявність атаки, зіставляючи вхідні і вихідні дані СВВ. Згідно з попереднім визначенням, сигнатурні методи будемо вважати верифікованими, а кластерні – не верифікованими. Верифікованість дозволяє провести експертну оцінку коректності методу і його реалізації в довільний момент часу, в тому числі в процесі експлуатації системи виявлення на його основі. Властивість верифікованості методу важливо при експлуатації системи виявлення атак в реальному середовищі в якості засобу збору доказової бази про атаки.

Можливі значення показника верифікованості:

– висока (+);

– низька (-).

Для розробки майбутнього програмного модуля планується високе значення рівня верифікованності.

*Адаптивність методу*. Оцінка стійкості методу до малих змін реалізації атаки, які не змінюють результат атаки. Адаптивність є єдиною суттєвою перевагою альтернативних методів виявлення атак перед традиційними сигнатурними. Відсутність адаптивності не дозволяє системі захисту оперативно реагувати на невідомі атаки і вимагає організації системи регулярного оновлення баз відомих атак, за аналогією з антивірусними системами.

Можливі значення показника адаптивності:

– висока (+);

– низька (-).

Для розробки майбутнього програмного модуля планується високе значення адаптивності за рахунок застосування технологій машинного навчання.

*Стійкість.* Ця характеристика описує незалежність вихідних даних методу від захищаємої системи - для одного і того ж входу метод повинен давати один і той же вихід, незалежно від захищаємої системи. Проблема стійкості особливо важлива для статистичних методів, які аналізують абсолютні значення параметрів продуктивності і завантаженості ресурсів мережі і вузлів, які можуть істотно відрізнятися на різних вузлах і в різних мережах. Навчений в одній мережі розпізнавач може бути стійким в межах даної мережі і нестійким у всіх інших мережах. Так як процедура навчання зазвичай є «дорогою» - вимагає використання великої кількості ресурсів і часу - число процедур навчання бажано мінімізувати. Методи виявлення атак, що аналізують семантику введення, більш стійкі, ніж статистичні.

Можливі значення показника стійкості:

– глобальна (+);

– локальна (-).

Для розробки майбутнього програмного модуля планується глобальний рівень стійкості за рахунок застосування технологій машинного навчання.

*Обчислювальна складність*. Теоретична оцінка складності методу. Цей критерій є ключовим для завдання виявлення атак в мережах і має набагато більше значення, ніж складність по пам'яті через випереджальне збільшення пропускної здатності каналів передачі даних і здешевлення машинної пам'яті.

Можливі види обчислювальної складності:

– сублінійна: константа, логарифм;

– лінійна;

– квадратична та ін.

В даному переліку не розглянуті також такі менш важливі критерії, як повнота і точність методу. [19]

## 2.2 Формальна модель веб-сторінки

Формально задача виявлення шкідливих сторінок зводиться до задачі класифікації, тобто розробки функції-гіпотези:

**hƟ(i)(x) = Pr[y = i | x,Ɵ], x ∈ P,** (3)

де **h: P → C** є проекцією множини ознак **P = {pj}** інтернет-сторінок на множину **C = { 'benign', 'redirect', 'exploit'}** класів, а **θ** – вектор параметрів моделі машинного навчання. При цьому **pi ∈ Rn** є вектором в n-вимірному просторі ознак інтернет-сторінки, **x ∈ P** – вектором ознак, а **y ∈ N** - числовим кодом класу. Клас ‘benign’ відповідає нешкідливим інтернет-сторінкам, ‘redirect’ - вузлам перенаправлення, ‘exploit’ - атакуючим сторінкам.

Для розробки моделі інтернет-сторінки пропонується використовувати теорію формальних мов, а саме - регулярні мови. [21] Таблиця 2 містить базові складові пропонованої моделі:

Таблиця 2 – Базові елементи формальної моделі інтернет-сторінки

|  |  |
| --- | --- |
| Елемент моделі | Опис |
| Алфавіт **Ξ = {0, 1, … , 255}** | Множина можливих числових представлень символу розміром 1 байт. |
| Документ **D ∈ L(Ξ\*)** | Рядок алфавіту **Ξ**, що представляє собою зміст веб-сторінки. |
| **T** | Множина рядків, відповідних усім припустимим тегам HTML. |
| **W** | Множина рядків, відповідних усім допустимим лексемам JavaScript. |
| Символ-пробіл **S** | Регулярний вираз, відповідний до будь-якого пробільного символу (пробіл, табуляція, новий рядок і т.д.) |
| Будь-який не пробільний символ **□** | Регулярний вираз, відповідний до будь-якого не пробільного символу. |
| JavaScript-сценарій **WD** | **L(‘<S\* script□\*></script’)** |

Таким чином, якщо інтернет-сторінку D як рядок, то регулярні вирази реалізують зручний механізм опису різних елементів сторінки, необхідних для обчислення значень ознак. Нові ознаки можуть проваджуватися у модель за рахунок створення регулярних виразів і визначення різних функцій над відповідними їм множинами рядків.

## 2.3 Нейронні мережі та їх використання у побудові імунних систем

Нейронна мережа є набором простих елементів, які називаються нейронами. [22] Нейрон являє собою лінійний автомат, який реалізує зважену суму декількох входів відповідно до набору вагових коефіцієнтів, а потім обчислює функцію Хевісайда або сігмоїдну функцію для отримання вихідного значення, що називається активацією нейрону. Вибір передавальної функції (*transfer function*) визначає, чи є нейрон бінарним або безперервно оціненим. Для формування нейронної мережі, ці нейрони з'єднані між собою відповідно до заданої топології. Ця топологія визначає вхідний шар, в якому активаційна функція нейронів встановлюється на вхідні значення, та вихідний шар, де читання активацій нейронів дає відповідь мережі. Тому нейронні мережі можуть розглядатися як нелінійна передавальна функція з одного векторного простору до іншого. [21] Вхідні ваги нейронів є параметрами цієї передавальної функції. Передавання вхідних даних до мережі спричиняє обчислення вихідних даних.

### 2.3.1 Принцип функціонування штучного нейрона

Штучний нейрон імітує властивості біологічного нейрона. На вхід штучного нейрона надходить деяка множина сигналів, кожен з яких є виходом іншого нейрона. Кожен вхід множиться на відповідний ваговий коефіцієнт, аналогічний синаптичній силі, і всі добутки підсумовуються, визначаючи рівень активації нейрона. На Рис. 2.1 представлена ​​модель, яка реалізує цю ідею. Хоча мережі бувають досить різні, в основі майже всіх їх лежить ця конфігурація. Тут безліч вхідних сигналів, позначених **x1, x2, ..., xn**, надходять на штучний нейрон. Ці вхідні сигнали відповідають сигналам, які приходять в синапси біологічного нейрона. Кожен сигнал множиться на відповідну вагу **w1, w2, ..., wn**, і надходить на підсумовуючий блок, позначений **Σ**. Кожна вага відповідає «силі» одного біологічного синаптичного зв'язку. Підсумовуючий блок, який відповідає тілу біологічного елемента, алгебраїчно об'єднує зважені входи, створюючи вихід **NET** (рис 2.1).

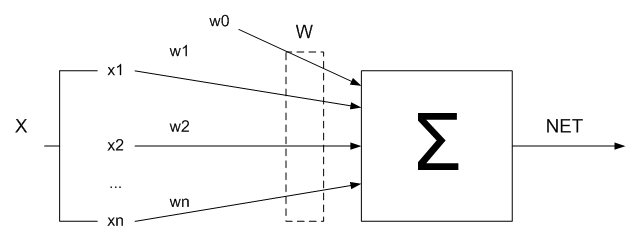


Рисунок 2.1 – Схема функціонування штучного нейрона

Даний опис можна представити у наступній формі:

**,** (4)

де **w0** – біас, **wi** – ваговий коефіцієнт i-го нейрона, **xi** - вихід i-го нейрона, **n** – кількість нейронів, що входять в оброблюємий нейрон. [26]

Сигнал **w0**, який має назву біас, відображає функцію граничного значення, зсуву. Цей сигнал дозволяє зсунути початок відліку функції активації, що в подальшому призводить до збільшення швидкості навчання. Цей сигнал додається до кожного нейрона, він вчиться як і всі інші вагові коефіцієнти, а його особливість в тому, що він підключається до сигналу +1, а не до виходу попереднього нейрона.

Отриманий сигнал **NET** як правило обробляється функцією активації і дає вихідний нейронний сигнал **OUT**, що зображено на Рис. 2.2:

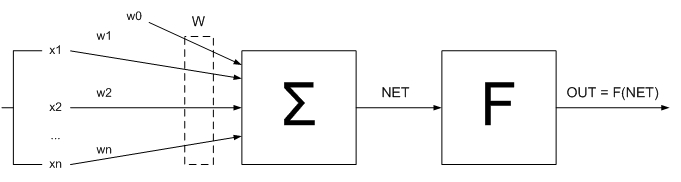


Рисунок 2.2 – Штучний нейрон з функцією активації

Якщо функція активації звужує діапазон зміни величини **NET** так, що при кожному значенні **NET** значення **OUT** належать деякому діапазону - кінцевому інтервалу, то функція **F** називається функцією, яка звужує. Часто використовується логістична або «сигмоїдальна» функція. Ця функція математично виражається наступним чином:

**,** (5)

Основна перевага такої функції - тe, що вона має просту похідну і диференціюється по всій осі абсцис. Функція підсилює слабкі сигнали і запобігає насиченням тяжкими сигналами [26]. Графік даної функції зображено на рис. 2.3.

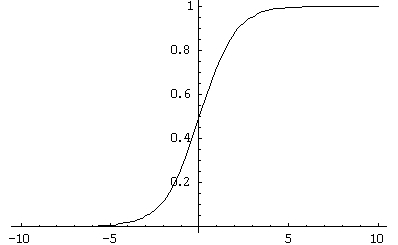


Рисунок 2.3 – Сигмоїдальна функція активації

Дуже важливою властивістю нейронних мереж є концепція програмування на прикладі. Велика кількість вагових коефіцієнтів ускладнює їх фіксування для отримання бажаного результату. Замість цього мережа програмується по прикладу та повторенню. Навчання здійснюється шляхом повторюваного подання пар «вхід-вихід». Кожен раз, коли представлено вхід, мережа прогнозує вихід. Вихідна частина пари «вхід-вихід» використовується для визначення, чи працює мережа на даний момент правильно чи ні. Якщо неправильно, то мережа коригується за допомогою алгоритму навчання, що базується на методі градієнту, застосованого до помилки виходу для зміни вагових коефіцієнтів. Після кожної модифікації мережа стає ближче до бажаної передавальної функції.

Потенціальні переваги цієї технології, як методу рішення задачі виявлення шкідливого коду, визначила Т.Ф. Лант (1990). [23] Harris Corporation у своєму проекті (1990) пропонує спосіб використання алгоритму нейронної мережі, що має назву «карта Кохонена», для виявлення вірусів у комп’ютерній системі. [24]

Підхід, що буде розглянуто далі у цьому розділі, базується на методі зворотного поширення помилки (*backpropagation method*).

Задачу, що у контексті даної курсової роботі повинна бути вирішена засобами нейронної мережі, можна виразити наступним чином:

– Вхідні дані: формальна модель веб-сторінки – тобто вхідний вектор, що містить x1, x2, … , xn, де xi – деякий елемент моделі, n – загальна кількість елементів.

– Вихідні дані: нейронна мережа має 3 виходи, кожен з яких відповідає класу веб-сторінки: *benign, redirect, exploit* (детальніше – див. розділ 2.2). Відповідь системи будується в залежності від того, який з трьох виходів має найбільше значення вихідного сигналу.

### 2.3.1 Властивості нейронних мереж

У рамках вирішення поставленої у п. 1.5 задачі потрібен інструмент, що навчається на базі тимчасових рядів. Тож застосування технологій нейронних мереж у цій системи є повністю оправданим. Мережа буде обробляти кожен зразок (*sample*) один раз і використовувати його для навчання. У цьому випадку рівномірність є дуже важливою, так як це означає, що мережа буде обробляти декілька зразків, що мають практично однаковий контент (для процесу навчання у нейронній мережі це є нормальним явищем).

Мережа адаптує свої вагові коефіцієнти для реалізації функції. Якщо існують кореляції у вхідному векторі, він буде автоматично приймати їх до уваги. Це допоможе процесу навчання сходитися і направить мережу до більш стабільного стану. Таким же чином, якщо вхід не має значення до проблеми, мережа нейтралізує вагові коефіцієнти, що надходять з нейронів, які підтримують цей вхід. Це допомагає визначити корисність вхідних даних, а також гарантує, що модель не буде порушена цими недоречними входами. Вагові коефіцієнти, створені додатковими входами, лише обчислювані, та не мають впливу на продуктивність моделі.

Нейронні мережі добре справляються з нечіткими або «зашумленими» даними. Дані, що оброблюються системою, можуть бути неповними, іноді в них не вистачає структурних елементів або точності вимірювання. Проте, нейронна мережа здатна впоратися з такою проблемою: ця технологія дозволяє знизити втрату точності на виході відносно до тієї втрати точності, яку мали вхідні дані.

### 2.3.2 Модель функціонування компоненту нейронної мережі у програмному модулі

Для системи виявлення вторгнень пропонується блок-схема, що зображена на рис. 2.4.

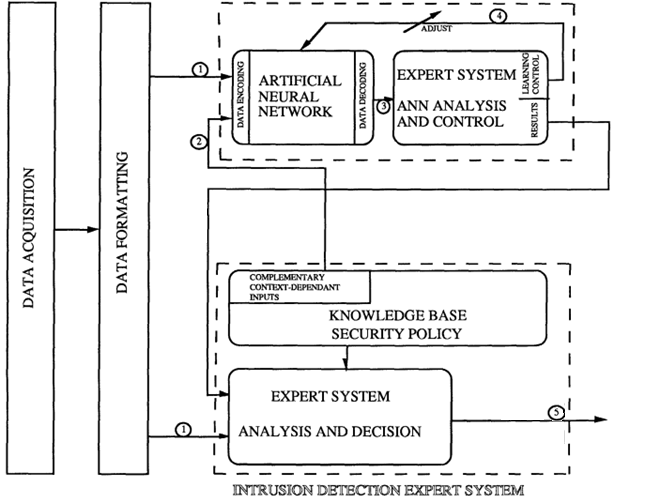


Рисунок 2.4 – Блок-схема функціонування СВВ

Схема, зображена на рис. 2.4, має наступні функціональні блоки:

– Data Acquisition (Збір даних).

Цей модуль збирає існуючі у системи дані аудиту та передає їх до системи виявлення вторгнень.

– Data Formating (Форматування даних).

Цей модуль конвертує дані у загальний формат, який може бути інтерпретований одночасно експертною системою і нейронною мережею. Він також перевіряє коректність вхідних даних.

– Artificial Neural Network (Штучна нейронна мережа).

Це нейронний двигун системи виявлення вторгнень, тобто сама нейронна мережа і функції, необхідні для оброблення формальної моделі веб-сторінки.

– Expert System: Neural Network Analysis and Control (Експертна система: Аналіз та управління нейронної мережі).

Цей модуль аналізує вихід нейронної мережі та переводить його в формат, придатний для виявлення вторгнень. Він також контролює внутрішні параметри мережі для отримання додаткової інформації. Відповідає за налаштування і перевірку процесу навчання, щоб уникнути зсуву системи з її моделі поведінки.

– Expert System: Analysis and Decision (Експертна система: Аналіз і формування рішення).

Цей модуль відповідає безпосередньо за виявлення вторгнень. Має базу знань, що містить відомі сценарії та недоліки ОС. Створює попередження і сигнали тривоги, якщо спостерігаються ознаки потенціальної загрози, виявлені за допомогою моделі нейронної мережі, або коли один з цих сценарії вторгнення спрацьовує.

Розбираючи надану схему поетапно, маємо наступну послідовність процесів:

1) отримання та форматування записів (сигнатур) з цільової системи;

2) обчислення додаткових контексто-залежних входів;

3) здобуття необробленого виходу нейронної мережі;

4) навчання з вчителем (*supervised learning*) та інтерпретація результатів;

5) формування остаточного рішення та генерація вихідного сигналу.

Подана модель використовує комбінацію чисельних методів (нейронні мережі) та символічне мислення (експертні системи) [25]. Як ми бачимо це зараз, нейронна мережа діє як фільтр для визначення, є аналізована веб-сторінка шкідливою чи ні. Швидкість аналізу значно зростає, так як у система працює на основі чисельних методів: тому очікується високий процент виявлення загроз та низька ймовірність хибних спрацьовувань. Решта вхідних даних, щодо яких система не може виявити явне рішення, передаються до експертної системи, що в свою чергу призводить до більш ефективної реалізації функціонування СВВ.

Таким чином, обрана найбільш ефективна модель машинного навчання, заснована на технології нейронних мереж, що демонструє істотно вищі показники, ніж більшість сучасних антивірусних сканерів.

# 3 РОЗРОБКА АРХІТЕКТУРИ програмного модулЮ для виявлення шкідливого коду

## 3.1 Онтологія проекту

Онтологія являє собою всеосяжну модель знань, яка дозволяє розробнику практикувати "високий" рівень повторного використання знань, пропонуючи більш детальну топологію між знанням і моделлю ПЗ в процесі розробки. [31]

Формалізовано онтологію деякої ПрО можна представити у вигляді впорядкованої трійки:

**О = <X, R, F>** (5)

де **X** - непорожня кінцева множина концептів (тобто сутностей, класів) ПрО, яку представляє онтологія **О**, **R** - кінцева множина відносин між концептами заданої предметної області, **F** - кінцева підмножина функцій семантичної (смислової) інтерпретації, заданих на множинах концептів **X** і відносин **R**. У найпростішому випадку, ці функції можуть являти собою логічно істинні висловлювання про властивості елементів цих множин.

Слід зазначити, що окремі концепти в описі онтології в загальному випадку представляють собою складні сутності, представлення яких передбачає використання більш простих класів. Тому для первинного опису онтологій доцільно використовувати лінгвістичний підхід, тобто представляти множини **X**, **R** і **F** у вигляді текстових специфікацій. При цьому важливо встановлювати правильні логічні відносини між окремими поняттями конкретної онтології. Для цього існують наступні лінгвістичні поняття і визначення:

1) Гіпонім – поняття, що в ставленні до іншого поняття виражає підвид, більш конкретне поняття. Відносно деякої множини об'єктів, гіпонім є поняттям, що відображає підмножину до початкового.

2) Гіперонім – поняття, що виражає більш загальну сутність в ставленні до іншої. Відносно деякої множини об'єктів, гіперонімом є поняття, що відображає надмножину до початкового.

3) Холонім - поняття, що виражає ціле щодо іншого поняття, в яке воно входить.

4) Меронім - поняття, щодо іншого поняття виражає складову частину.

Для поданої ПрО було розроблено наступний перелік концептів:

1. **Проактивний захист** (Proactive Security) - сукупність технологій і підходів, що використовуються в антивіруснjve ПЗ, метою якого є запобігання зараженню системи шкідливим ПЗ.

*Гіпонім*: Віртуалізоване Робоче Середовище.

2. **Віртуалізоване Робоче Середовище** - технологія, яка працює за допомогою системного драйвера, який перехоплює всі запити запису до системи. Замість того, щоб здійснювати запис на жорсткий диск, він записує дані в буфер (спеціальний простір на диску).

*Гиперонім*: проактивний захист.

3. **Формальна Модель Веб-Сторінки** (WPFM) - опис колекції елементів веб-сторінок за допомогою регулярної мови.

*Меронім*: Елемент Веб-Сторінки.

4. **Елемент WPFM** - основний компонент Формальної Моделі Веб-Сторінки.

*Холонім*: Формальна Модель Веб-Сторінки.

5. **Веб-сторінка** - інформаційний ресурс, який можна переглянути у веб-браузері. Зазвичай, інформація веб-сторінки записана в форматі HTML, XHTML.

*Холонім*: Вибірка Веб-Сторінок.

*Гіпонім:* MWP.

5. **Шкідлива Веб-Сторінка (MWP)** - веб-сторінка, що містить деякий потенціально шкідливий JavaScript-сценарій.

*Меронім*: Шкідливий код.

*Гіперонім:* Веб-сторінка.

6. **Шкідливий код** – ділянка коду в JavaScript-сценарії, розташованому на веб-сторінці, яка реалізує шкідливий для файлів або даних користувача механізм.

Холонім: Шкідлива Веб-Сторінка.

7. **Вибірка Веб-Сторінок** – деяка вибірка веб-сторінок для навчання системи, що містить шкідливі та безпечні веб-сторінки.

Меронім: Шкідлива Веб-Сторінка.

Холонім: Інтелектуальна Система Виявлення Шкідливих Веб-Сторінок.

**8. Система Виявлення Шкідливого Коду (MCDS) –** програмний модуль для виявлення потенційно шкідливих ділянок програмного коду.

*Гіпонім:* MWPDIS*.*

**9. Інтелектуальна Система Виявлення Шкідливих Веб-Сторінок (MWPDIS) –** програмний модуль для виявлення шкідливих веб-сторінок, функціонуючий за допомогою технології машинного навчання.

*Гіперонім: MCDS.*

*Меронім: Вибірка Веб-Сторінок.*

Певний логічний фрагмент поданого лінгвістичного опису (включаючи концепти 3 – 9) представлений за допомогою нотації UML на рисунку 3.1. Більш детальне схематичне відображення фрагменту онтології проекту показане на рисунку 3.2.

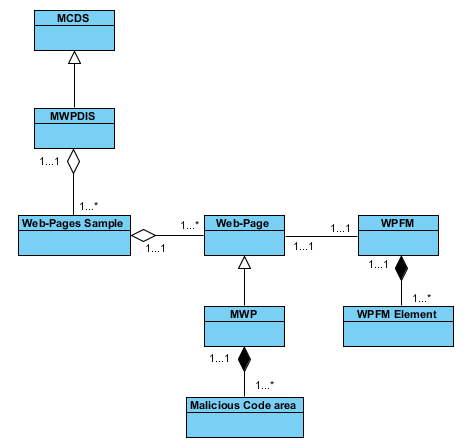


Рисунок 3.1 – Схема логічного фрагменту онтології в нотації UML

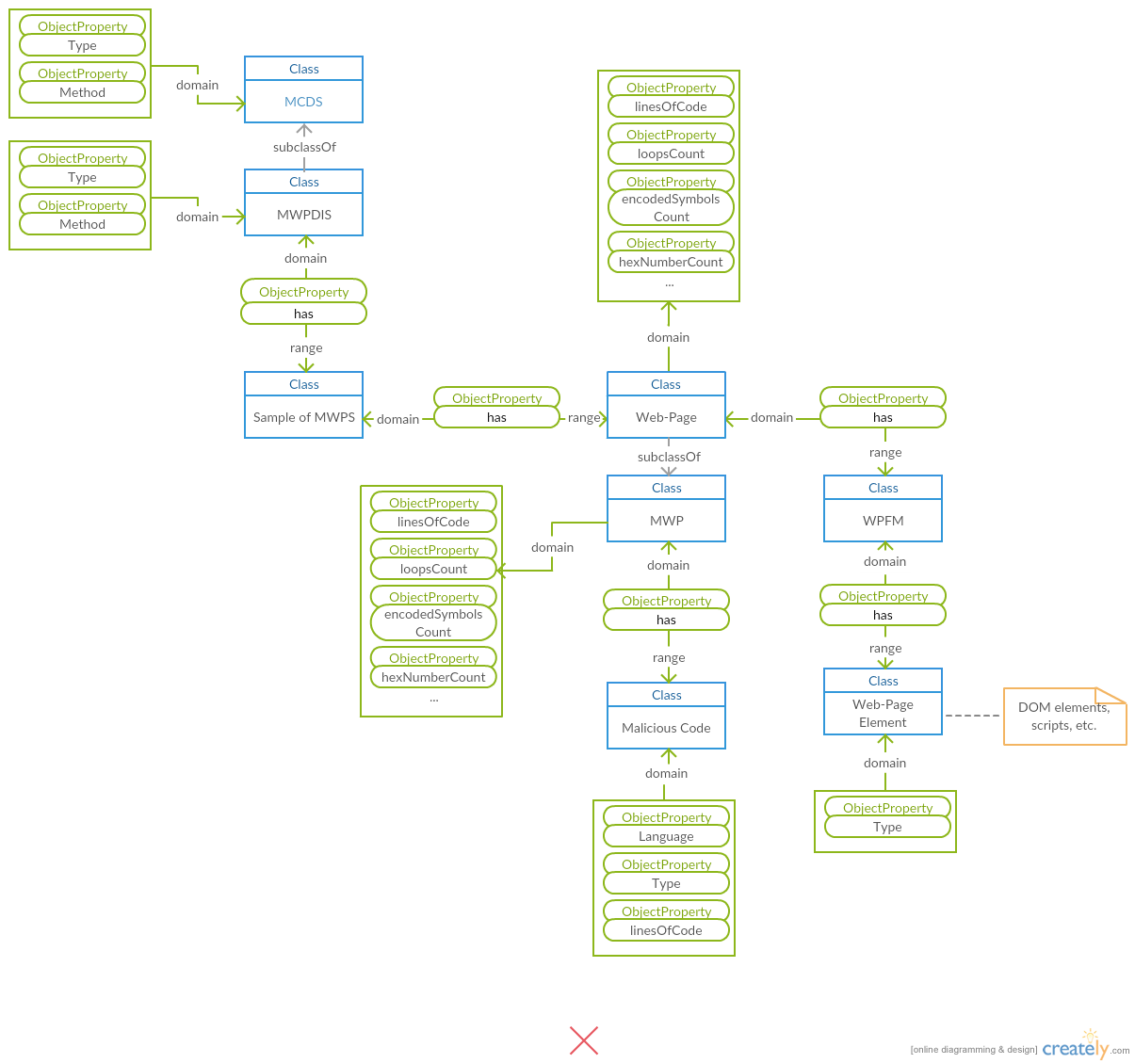


Рисунок 3.2 – Деталізоване зображення фрагменту онтології проекту

## 3.2 Бізнес-логіка проекту

Було розроблено наступний перелік первинних бізнес-правил проекту:

1) Інтелектуальна Система Виявлення Шкідливих Веб-Сторінок (MWPDIS) повинна мати можливість обробити будь-яку веб-сторінку, що відповідає стандартам The World Wide Web Consortium. [32]

2) Якщо користувачем було зроблено запит на аналіз веб-сторінки, що не можете буде представлена у формі WPFM, або поданий до системи файл не є веб-сторінкою, система повинна відреагувати виводом помилки.

3) Якщо система по деяким причинам виявляє клас поданої веб-сторінки неправильно, користувачу буде запропоновано вказати її тип власноруч, тим самим «довчити» інтелектуальну систему в процесі функціонування.

4) Процес навчання здійснюється за допомогою окремого модулю, що приймає вибірку веб-сторінок як вхідні дані та обробляє їх по принципу «навчання з вчителем» (див. розділ 2.3.2).

5) Подана системі вибірка Веб-сторінок повинна містити хоча б одну повністю безпечну сторінку та хоча б одну шкідливу.

6) Шкідлива сторінка вважається таковою, якщо хоча б один з JavaScript-сценаріїв, що міститься в неї, є шкідливим.

7) JavaScript-сценарій вважається шкідливим, якщо він містить хоча б одну ділянку шкідливого коду.

## 3.3 Діаграма прецедентів

Діаграму основних прецедентів майбутнього програмного модулю зображено на рис. 3.3.

Основний функціонал складається з двох прецедентів. Навчання системи (*Teach the system*) включає до себе завантаження експериментальної вибірки веб-сторінок (*Load experimental data*) та формування Формальної Моделі для кожної веб-сторінки (*Construct WPFMs*), формування нейронною мережею виходу (*Construct NN output*), аналіз виходу нейронної мережі (*Analyze NN output*). Аналіз певної веб-сторінки (*Analyze a particular web-page*) у свою чергу включає до себе формування Формальної Моделі веб-сторінки (*Construct WPFM*), здобування відповіді від системи (*Get answer from the system*); розширенням цього прецеденту є можливість довчити систему (*Perform additional teaching of the system*).

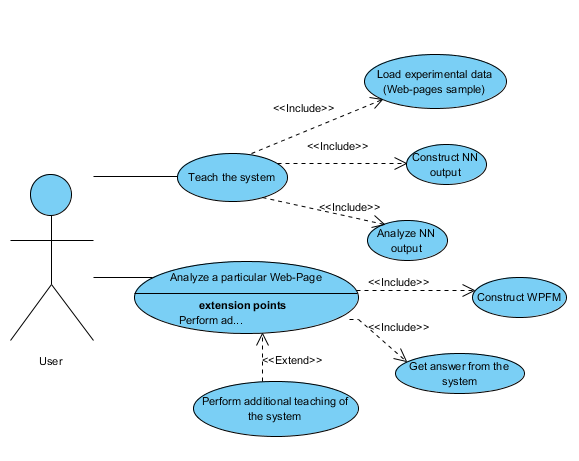


Рисунок 3.3 – Діаграма прецедентів

## 3.4 Діаграма IDEF0

Основна функція системи, якою є аналіз веб-сторінки на предмет шкідливості, відображена у форматі діаграми IDEF0 на рис. 3.4 та рис. 3.5.

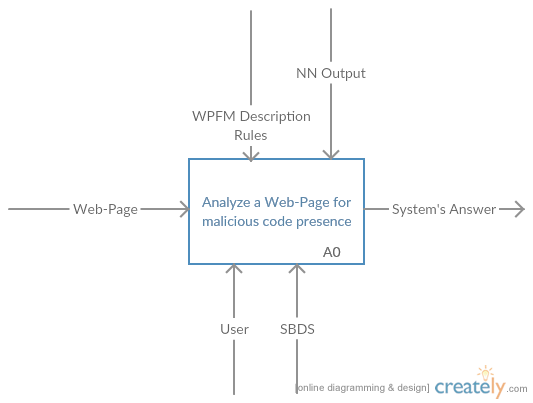


Рисунок 3.4 – Діаграма IDEF0 (Context Level)

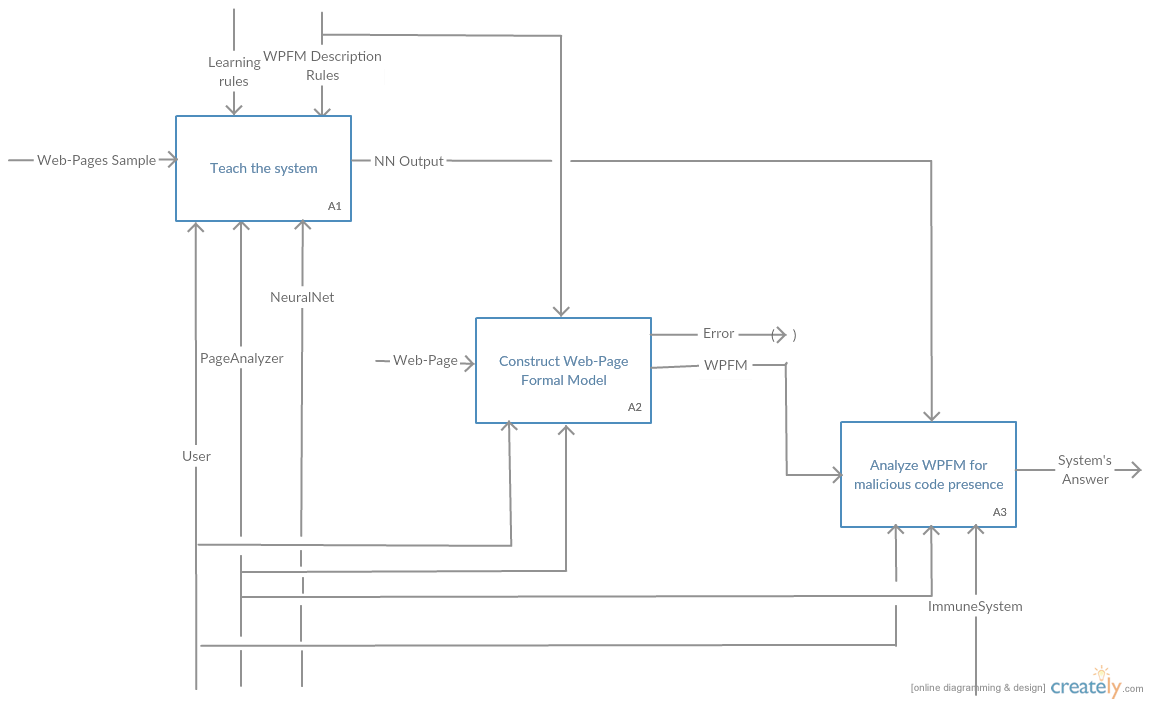


Рисунок 3.5 – Діаграма IDEF0 (Decomposition Level)

Перший блок діаграми А1 відображає процес навчання системи. Вхідними даними для цього блоку є вибірка веб-сторінок, вихідними – вихід нейронної мережі. Управління здійснюється за допомогою алгоритму навчання системи та правил конструювання WPFM. Механізмами на цьому етапі є користувач та два модулі *PageAnalyzer* і *NeuralNet*.

Другий блок діаграми А2 відображає процес репрезентування коду веб-сторінки у Формальну Модель Веб-сторінки. Вхідними даними для цього блоку є сама веб-сторінка, вихідними – її Формальна Модель чи повідомлення про помилку. Управління здійснюється за рахунок правил конструювання WPFM, а механізмами на цьому етапі є користувач та програмний модуль *PageAlalyzer.*

Третій блок діаграми А3 відображає наступний процес аналізування веб-сторінки на предмет присутності шкідливого коду. Вхідними даними є скомпонована на попередньому етапі WPFM модель. Обидва попередні механізми також присутні на цьому етапі, також добавляється другий програмний модуль *ImmuneSystem*. Управління здійснюється за рахунок статистичних даних, що містяться у базі даних. Виходом на цьому етапі є відповідь системи на питання, чи є сторінка потенційно шкідливою.

## 3.5 Проектування архітектури системи

### 3.5.1 Діаграма розгортання компонентів

Для побудови архітектури системи було обрано архітектурну модель типу «Клієнт-Сервер» з «тонким» клієнтом.

Основними перевагами цієї архітектурної моделі є:

– зменшення обсягу обчислювального навантаження на клієнтські вузли, що означає підвищення рівня масштабованості всієї системи;

– зниження витрат на супровід ПЗ, тому що тепер компоненти BLS розміщуються на сервері і можуть супроводжуватися централізовано, і їх змінена функціональність буде відразу ж доступна всім клієнтським застосуванням;

- Знижується мережевий трафік, тому що в такій системі квантом обміну між клієнтським вузлами з сервером БД є вже не файл даних, а SQL-запит на деяку обробку (вибірку) даних і відповідний результат (record set) [34];

– На клієнтських вузлах функціонують тільки компоненти PRS, тобто вони реалізують тільки операції введення-виведення даних, а також, можливо, найпростіші сервісні функції, напр. первинну валідацію вхідних даних, зміна форми подання вихідних даних і т.д. [35].

Діаграма розгортання компонентів зображена на рис. 3.6.

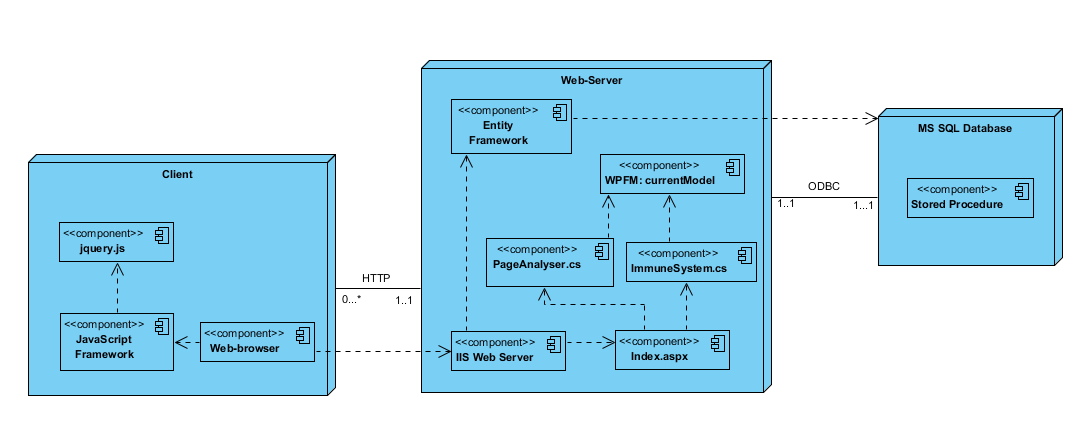


Рисунок 3.6 – Діаграма розгортання компонентів

Вузол «*Client*» має наступні компоненти:

– *JavaScript Framework* – компонент, необхідний для реалізації необхідної бізнес-логіки на стороні клієнта.

– *jquery.js* – бібліотека JavaScript, необхідна для реалізації користувацького інтерфейсу.

– *Web-browser* – програма – веб-браузер, що надає користувачу доступ до системи.

Клієнт взаємодіє з веб-сервером за допомогою протоколу HTTP (конкретним веб-сервером, що планується до застосування у майбутньому проекті, є сервер Microsoft IIS).

Вузол «*Web-Server*» має наступні компоненти:

*– Index.aspx* – головна веб-сторінка додатку.

*– ImmuneSytem.cs, PageAnalyser.cs* – програмні модулі, які реалізують необхідний функціонал системи (функціонування імунної системи на основі нейронної мережі та сканування веб-сторінок відповідно).

– *WPFM: currentModel* – веб-сторінка, що аналізується системою, репрезентується у вид Формальної Моделі Веб-сторінки (WPFM). Об’єкт WPFM створюється у момент, коли користувач робить запит на сканування деякої веб-сторінки.

– *Entity Framework* – програмний компонент, що реалізує об’єктно-реляційне відображення (object-relational mapping) при роботі з базою даних.

Вузол *Database,* який буде розроблено за допомогою СКБД MS SQL, буде зберігати статистичні дані про результати навчання системи, необхідні для її функціонування. Цей вузол містить у собі компонент *Stored Procedure* – це є як мінімум одна збережена процедура, що реалізує додаткову бізнес-логіку системи на рівні бази даних.

Комунікація веб-серверу з базою даних здійснюється засобами драйверу ODBC.

### 3.5.2 Вибір архітектурних патернів

Архітектурні патерни, вибрані для реалізації програмної системи, перелічені у табл. 3:

Таблиця 3 – Обрані архітектурні патерни

|  |  |  |
| --- | --- | --- |
| *Назва патерну* | *Опис* | *Обґрунтування* |
| **Фасад** | Надає єдиний інтерфейс до безлічі операцій або інтерфейсів в системі на основі уніфікованого інтерфейсу для полегшення  роботи з системою. | У проекті очікується наявність великої кількості функцій, що реалізують сканування вихідного коду веб-сторінки. Об’єднання цих функцій в один загальний інтерфейс підвищить ремонтопридатність (*maintainability*) коду. |
| **Ітератор** | Дає можливість послідовно перебрати  всі елементи складеного об'єкта, не розкриваючи  його внутрішнього представлення. | WPFM (Формальна Модель Веб-сторінки) є складним об’єктом, що репрезентує множину елементів та характеристик веб-сторінки. Коли система сканує деяку веб-сторінку, відповідний модуль повинен ітеративно обробити кожен елемент множини, - що й об*ґ*рунтовує використання цього патерну. |

# Висновки

В ході даної курсової роботи було:

* проаналізовано предметну область індивідуальної задачі;
* досліджено існуючі методи детектування шкідливого ПЗ;
* досліджено існуючі види шкідливого ПЗ як програмні артефакти;
* досліджено деякі відкриті системи виявлення шкідливого ПЗ та їх характеристики;
* розглянуто та проаналізовано характеристики та параметри системи виявлення шкідливого ПЗ, побудовано відповідну математичну модель;
* розроблено модель інтернет-сторінки на основі формальних мов, побудовано її математичну модель;
* досліджено та проаналізовано алгоритми виявлення шкідливих інтернет-сторінок на основі технологій машинного навчання, розглянуто математичну модель функціонування нейронної мережі у контексті поставленої задачі;
* проведено дослідження моделі нейронної мережі для вирішення задачі виявлення шкідливих інтернет-сторінок.
* розроблено первинну онтологію та бізнес-логіку проекту;
* побудовано UML-діаграми, що відображають опис прецедентів, основний функціонал та майбутню архітектуру системи.

Отже, в ході виконання даної курсової роботи було проведено дослідження всіх необхідних теоретичних аспектів і проаналізовано та побудовано необхідні математичні моделі для подальшої розробки програмного модулю для виявлення шкідливих веб-сторінок з використанням технології машинного навчання, було розроблено та задокументовано опис його майбутньої архітектури.

# Список джерел інформації

1. Дасгупта В. Искусственные иммунные системы. М.: Физматлит, 2006. – 344 c.
2. Mihai Christodorescu, Somesh JhaTesting. Malware Detectors. Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA’04), Boston, Massachusetts, USA, July 11-14, 2004, 11 p.
3. Сигнатура атаки – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BD%D0%B0%D1%82%D1%83%D1%80%D0%B0\_%D0%B0%D1%82%D0%B0%D0%BA%D0%B8,01.10.16
4. McAfee AVERT. Virus information library – Режим доступу до ресурсу: http://us.mcafee.com/virusInfo/default.asp, 01.10.16
5. Symantec Antivirus Research Center. Expanded threat list and virus encyclopedia. – Режим доступу до ресурсу: http://securityresponse.symantec.com/avcenter/venc/data/cih.html, 01.10.16
6. Snort 2.1. Обнаружение вторжений / Джей Бил. и др. – М.: ООО «Бином-Пресс», 2006 г. – 656 с.
7. A comprehensive tutorial on cross-site scripting – Режим доступу до ресурсу: https://excess-xss.com/, 09.10.16
8. Michael Sikorski and Andrew Honig. Practical Malware Analysis. No Starch Press, Inc., 2012.
9. Microsoft Corporation. Naming malware – Режим доступу до ресурсу: http://www.microsoft.com/security/portal/mmpc/shared/malwarenaming.aspx, 15.10.16
10. Peter Szor. The Art of Computer Virus Research and Defense. Addison Wesley Professional, February 2005.
11. PandaLabs. Quarterly Report; April-June 2014 – Режим доступу до ресурсу: http://www.pandasecurity.com/mediacenter/src/uploads/2014/07/Quarterly-Report-PandaLabs-April-June-2012.pdf, 22.10.16
12. John Aycock. Computer Viruses and Malware. Springer, 2006.
13. Vesselin Bontchev. Possible Virus Attacks Against Integrity Programs and How to Prevent Them. https://bontchev.nlcv.bas.bg/papers/attacks.html, 13.11.16
14. Как определяется предмет и объект исследования – http://www.zpu-journal.ru/asp/matriculation/faq/object/, 20.11.16
15. The Bro Network Security Monitor – https://www.bro.org/, 22.11.16
16. OSSEC: Open Source HIDS SECurity – http://ossec.github.io/about.html,
17. Prelude Universal Open-Source SIEM project – https://www.prelude-siem.org/
18. Snort: Network Intrusion Detection & Prevention System – https://www.snort.org/
19. Р.Л. Смелянский, Д. Ю. Гамаюнов. “Современные некоммерческие средства обнаружения атак”. // Факультет Вычислительной Математики и Кибернетики, МГУ им. М. В. Ломоносова, Москва, 2002 г.
20. Формальні мови та автомати // К.: Національний технічний університет України “Київський політехнічний інститут”, 2010. – 98 с.
21. Herve Debar, Monique Becker, and Didier Siboni, “A neural network component for an intrusion detection system.” // Proceedings of the 1992 IEEE Computer Sociecty Symposium on Research in Security and Privacy, Oakland, CA, USA, May 1992
22. An Introduction to computing with neural nets, Richard P. Lippmann, IEEE ASSP Magazine, April 1987, pages 4-22.
23. IDES: An Intelligent System for Detecting Intruders, Teresa F. Lunt, Proceedings of the symposium: Computer Security, Threat and Countermeasures,Rome, Italy, Nov 1990.
24. A Neural Network Approach Towards Intrusion Detection, Kevin L. Fox, Ronda R. Henning, Jonathan H. Reed, Richard P. Sitnonian, Harris Corporation, Government Information Systems Division, P.O. Box 98000, Melbourne, FL 32902, July 1990.
25. Reasoning System – https://en.wikipedia.org/wiki/Reasoning\_system
26. Ф. Уоссермен. Нейрокомпьютерная техника: Теория и практика. Перевод на русский язык Ю. А. Зуев, В. А. Точенов, 1992
27. Ian Sommerville SoftwareEngineering / 9edition. - Addison Wesley, 2011
28. Current UML Specification Version 2.5 – http://www.uml.org/
29. System Engineering Fundamentals — Defense Acquisition University Press, 2001 https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-885j-aircraft-systems-engineering-fall-2005/readings/sefguide\_01\_01.pdf
30. Karl E. Wiegers “Software Requirements (Second Edition)” –http://ab.kh.ua/books/Vigers%20Karl%20-%20Razrabotka%20Trebovany%20K%20Programmnomu%20obespecheniu.pdf
31. Ontology Modeling Using UML http://www.ucalgary.ca/wangx/files/wangx/oois.pdf
32. Ontological Context-awareness for Adaptive Augmented Reality // Ramón Hervas, José Bravo, Alberto Garcia-Lillo, Castilla-La Mancha University
33. The World Wide Web Consortium – https://www.w3.org/
34. Конолли Т. и др. Базы данных: проектирование, реализация и сопровождение. – М. «Вильямс», 2003. – 1120 с.
35. «Разработка и сопровождение проблемно-ориентированных ПС» / проф. Н.В. Ткачук / каф. МСиТ, ХНУ им. В.Н. Каразина