МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ С.В. Казмірчук
«_____»_____20__р.

На правах рукопису
УДК 004.056.5:510.22(043.3)

ДИПЛОМНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

Тема: Програмний модуль моніторнигу вразливостей GRID-систем

Виконавець: М.Б. Горова
Керівник: к.т.н. О.О. Висоцька
Нормоконтролер: к.т.н. О.О. Висоцька

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Бакалавр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ С.В. Казмірчук

«___»_____20__р.

ЗАВДАННЯ

на виконання дипломної роботи

здобувача вищої освіти Горової Маргарити Борисівни

1. Тема: *Програмний модуль моніторингу вразливостей GRID-систем* затверджена наказом ректора від «26» квітня 2021 р. №652/ст.
2. Термін виконання: з 10.05.2021 по 20.06.2021 р.
3. Вихідні дані: дослідити основні поняття вразливості, методи виявлення вразливостей; дослідити поняття GRID-систем; на основі проведеного аналізу розробити програмний модуль моніторингу вразливсотей GRID-систем.
4. Зміст пояснювальної записки: аналіз інсуючих вразливостей GRID-систем та методів їх виявлення, розробка та дослідження розробленого програмного модуля, що дозволяє отримати експерту характеристику вразливостей для того, аби полегшити та пришвидшити майбутній аналіз ризиків.

КАЛЕНДАРНИЙ ПЛАН

виконання дипломної роботи

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	19.04.2021	*Виконано*
2.	Аналіз літератруних джерел	20.04.2021	*Виконано*
3.	Обгрунтування рішення	27.04.2021	*Виконано*
4.	Збір інформації	28.04.2021	*Виконано*
5.	Аналіз понять GRID-систем	07.05.2021	*Виконано*
6.	Аналіз та дослідження вразливостей	10.05.2021	*Виконано*
7.	Розробка та опис методики виявлення вразливостей	14.05.2021	*Виконано*
8.	Реалізація та дослідження власного програмного модуля моніторингу вразливостей GRID-систем	20.05.2021	*Виконано*
9.	Оформлення та друк пояснювальної записки	30.05.2021	*Виконано*
10.	Оформлення презентації	31.05.2021	*Виконано*
11.	Перевірка на антиплагіат	05.06.2021	*Виконано*
12.	Отримання рецензій від рецензентів	10.06.2021	*Виконано*
13.	Підготовка до захисту	14.06.2021	*Виконано*

Здобувач вищої освіти М.Б. Горова

(підпис, дата)

Керівник дипломної роботи О.О. Висоцька

(підпис, дата)

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох рохділів, загальних висновків, списку використаних джерел, додатків, загальним обсягом робота складає 112 сторінок, має 23 рисунки, 1 таблицю, 1 додаток. Список використаних джерел містить 32 найменування і займає 3 сторінки.

Метою дипломної роботи є розробка програмного модулю моніторнигу вразливості GRID систем.

В дипломній роботі розглянуті питання дослідження вже відомих систем та програмних рішень моніторнигу вразливсотей GRID систем. Розроблено алгоритм та програмну реалізацію модулю моніторнигу вразливості GRID систем. Проведено тестування та оцінка доцільності використання розробленого програмного модулю моніторнигу вразливості GRID систем.

Запропонований програмний модуль моніторнигру вразливостей GRID-систем дозволяє забезпечити захист інформації при її передачі в інформаційних мережах.

Ключові слова: вразливості GRID-систем, моніторинг та виявлення атак, аналіз вразливості, робота мережі, програмне забезпечення.

ЗМІСТ

ВСТУП 6
РОЗДІЛ 1 ВРАЗЛИВОСТІ GRID-СИСТЕМ 9
1.1 Особливості GRID-систем 9
1.2 Аналіз існуючих вразливостей 22
1.3 Оцінка кібербезпеки SMART GRID-систем 39
РОЗДІЛ 2 СУЧАСНІ РІШЕННЯ В СФЕРІ МОНІТОРИНГУ 45
2.1 Характеристика сучасних методів моніторингу та виявлення атак 45
2.2 Класифікація засобів моніторингу та аналізу 53
2.3. Методи оцінки критичності вразливостей 68
2.4. Порівняльна характеритсика програмних засобів на ринку 70
РОЗДІЛ 3 РОЗРОБКА АЛГОРИТМУ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ ТА КОНТРОЛЮ 73
3.1 Вибір мови програмування 73
3.2 Розробка алгоритму роботи програмного забезпечення 74
3.3 Розробка інтерфейсу програми 76
3.4 Опис та тестування розробленого програмного забезпечення 77
ВИСНОВКИ 80
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ: 81
ДОДАТОК А 84

ВСТУП

Актуальність. Термін «GRID» був придуманий Яном Фостером та Карлом Кессельманом, авторами першої книги про ідею використання комп'ютерних мереж для вирішення проблем, що вимагають особливо великих обчислювальних ресурсів.

Метою створення Grid є інтеграція набору просторово розподілених ресурсів, щоб мати можливість запускати широкий спектр програм на будь-якому наборі цих ресурсів, незалежно від їх розташування.

Глобальні Grid обчислювальні мережі запропоновано як нову парадигму для вирішення масштабних обчислювальних проблем у науці, техніці та бізнесі. Вони дозволяють одночасно використовувати мільйони обчислювальних ресурсів, що належать різним організаціям та знаходяться в різних адміністративних районах. Grid системи поєднують неоднорідні обчислювальні ресурси (персональні комп'ютери, робочі станції, кластери, суперкомп'ютери), використовуючи різні стратегії доступу, виконуючи різні програми (наукові, інженерні та комерційні), які ставлять різні вимоги до системи. Ресурси належать різним організаціям, які мають власну політику управління, використання та оцінки ресурсів для різних користувачів у різний час. Доступність та завантаженість ресурсів також може динамічно змінюватися в часі.

В мережевому середовищі власники ресурсів та споживачі мають різні цілі та використовують різні стратегії та економічні схеми для коригування попиту та пропозиції. Тому нагальним питанням є розробка системи управління мережевими ресурсами, яка спрямована на оптимізацію відносин між власниками ресурсів та користувачами, відповідно до обраної ними стратегії. Багато систем управління мережевими ресурсами (такі як Legion, Condor, AppLeS PST, NetSolve, PUNCH, XtremWeb і т.д.) використовують самі прості схеми розподілу, в яких компоненти, що відповідають за розподіл, використовують функції витрат, визначені системними параметрами, щоб визначити, які завдання слід

виконувати над якими ресурсами. Метою цієї системи розподілу являється збільшення пропускної системи, завантаження та скорочення часу виконання завдань, а не збільшення рентабельності ресурсів та додатків. Вони не враховували вартість використання кожного ресурсу, а це означає, що всі додатки мають однакову важливість у будь-який час, але насправді це далеко не так – важливість повинна зростати в міру наближення завдань програми. Коли рішення про розподіл ресурсів приймаються динамічно і залежать від поточних потрем користувачів, для планування та розподілу ресурсів рекомендується використовувати економічні методи. Це ринкова модель розподілу ресурсів, і ціна кожного ресурсу визначається потребами користувачів та їх доступністю. Тому в Grid системі користувачі конкурують з іншими користувачами, а власники ресурсів конкурують з іншими власниками ресурсів.

Економічний підхід дозволяє успішно керувати розпоширеними та неоднорідними ресурсами, як це відбувається в реальній економіці. Економічні системи управління ресурсами Grid динамічно визначають найкращі ресурси, при цьому враховуючи їх ціну і продуктивність, і розподіляють завдання на цих ресурсах так, щоб задовольнити потреби користувачів.

Метою роботи є розробка програмного модулю моніторингу вразливості GRID систем.

Виходячи з мети, **завданням** даної роботи є:

1. провести аналіз існуючих вразливостей, а також відомих систем та програмних рішень моніторингу вразливостей GRID систем;

2. розробити алгоритм та програмну реалізацію модулю моніторингу вразливості GRID систем;

3. провести тестування розробленого програмного модулю моніторингу вразливості GRID систем.

Галузь застосування. Розроблений програмний модуль відносяться до галузі інформаційної безпеки і може бути використаним для підвищення рівня захищеності мережі за рахунок нейромережевого методу обробки пакетів та балансування навантаження.

Об’єктом дослідження є процес моніторингу вразливостей GRID системи.

Предметом дослідження є методи виявлення атак та моніторингу вразливості GRID систем.

Методи досліджень. Проведені дослідження базуються на сучасних методах та засобах виявлення мережевих атак і моніторингу вразливості GRID систем.

Практична цінність. Розроблений програмний модуль забезпечує виконання моніторингу вразливостей GRID систем на основі загальної системи оцінки вразливостей (CVSS), що дозволяє здійснювати аналіз вразливостей системи для запобігання подальшого їх використання з метою порушення безпеки системи.

РОЗДІЛ 1. ВРАЗЛИВОСТІ GRID-СИСТЕМ

1.1. Особливості GRID-систем

Сьогодні людей дуже турбує розвиток високопродуктивних обчислнь, які підтримують цю сферу. Це пояснюється тим, що сучасній науці та техніці потрібно вирішити велику кількість проблем. У рамках існуючих наукових експериментів та методів моделювання вирішення цих проблем вимагає великих обчислювальних потужностей, тому в багатьох наукових галузях традиційні методи аналізу вичерпали свою силу. Технологія Grid використовується для забезпечення високої продуктивності для вищезазначених практичних завдань.

Технологія Grid використовується для створення географічно розподіленої обчислювальної інфраструктури, яка поєднує різні типи ресурсів із колективним досутпом до цих ресурсів у віртуальній організації [1]. Ця технологія може бути використана для вирішення науково-математичних задач, для вирішення яких потрібно багато обчислювальних ресурсів. Grid обчислення також використовуються в комерційній інфраструктурі для вирішення трудомістких завдань, таких як економічне прогнозування, сейсмічний аналіз, а також розробка та дослідження нових характеристик ліків.

Термін «Grid» використовується з середини 1990-х років і був обраний за аналогією з енергетичними мережами [1]. Розробка та впровадження мережевих технологій має стратегічне значення. Найближчим часом ця технологія створить новий обчислювальний інструмент для розвитку високих технологій у різних сферах людської діяльності.

Ідейною основою Grid технологія є об’єднання ресурсів шляхом створення нового типу комп’ютерної інфраструктури, що забезпечує глобальну інтеграцію інформаційних та обчислювальних ресурсів на основі мережевих технологій та спеціального проміжного програмного забезпечення (між базовим та прикладним програмним забезпеченням) та набір стандартизованих служб для

забезпечення надійного спільного доступу до географічно розподіленої інформації та обчислюваних ресурсів: персональних комп'ютерів, кластерів, сховищ інформації та мереж.

Поява технології Grid зумовлена наступними факторами:

– потребує вирішення складних наукових, промислових, інженерних та комерційних проблем;

– швидкий розвиток мережевого середовища передачі та високошвидкісної технології передачі даних;

– наявність обчислювальних ресурсів у багатьох організаціях: суперкомп'ютери, або найчастіше, організовані в кластери персональних комп'ютерів.

Використовуючи технологію Grid – ви можете забезпечити новий високоякісний рівень, а в деяких випадках ви навіть можете застосувати новий метод для обробки великих обсягів експериментальних даних, забезпечити складне моделювання процесів, візуалізацію великого набору даних, з великим обчисленням обсягу [3]. На сьогоднішній день реалізовано та реалізується багато проектів для створення мережевої системи. Більшість із цих проектів є експериментальними [3].

Згідно з результатами аналізу проекту, ми можемо узагальнити три напрямки розвитку Grid технології: обчислювальні Grid, щільні Grid обробки даних та семантичні Grid для експлуатаційних даних з різних баз даних. Завдяки глобальному розподілу цих обчислень серед комп'ютерів, перший напрямок спрямований на досягнення максимальної швидкості обчислення. Як приклад цього напрямку може бути використаний проект DEISA, де робиться спроба об'єднати суперкомп'ютерний центр.

Мета другого напряму – використання відносно простої програми для обробки великих обсягів даних підповідно до принципу «одне завдання – один процесор». В такому випадку передача даних для обробки та надсилання результатів є досить складним завданням. Для цієї області мережева інфраструктура є кластером.

Семантична GRID (Semantic Grid) – відображає інфраструктуру для виконання обчислювальних завдань на основі розподіленого метаінформаційного середовища. Це дозволяє маніпулювати даними з різних типів баз даних та різних форматів та надавати результати у визначеному форматі програми.

Одним із проектів, спрямованих на створення виробничої Grid системи для обробки великих обсягів наукових даних, є проект EGEE (Enabling Grids for E-SCIENCE), який реалізується за підтримки Європейського Союзу. Проект EGEE – будівництво інфраструктури Grid головним чином зосереджується на її використанні в різних галузях досліджень, включаючи Європейський центр ядерних досліджень (CERN) LHC-прискорювач. Проект EGEE тісно пов'язаний з проектом LCG (LHC Computing Grid) на цьому етапі розвитку, який фактично є його технічною основою.

Концепція Grid

Технологія Grid забезпечує гнучкий, безпечний та скоординований доступ до ресурсів. У цьому випадку роль ресурсів може виконувати апаратне забезпечення (жорсткі диски, процесори) або системне та прикладне програмне забезпечення (бібліотеки, програми). З точки зору мережі, існує поняття «віртуальна організація», що позначає групу організацій та людей, які спільно вирішують загальні проблеми та надають ресурси один одному. Наприклад, віртуальна організація може бути колекцією всіх людей, які беруть участь у певній науковій співпраці. Склад, обсяг, тривалість, діяльність, цілі та взаємозв'язки між учасниками віртуальних організацій можуть відрізнятися. Склад віртуальної організації може динамічно змінюватися.

Існує два основних критерії, що відрізняють Grid системи від інших систем, що забезпечують розподілений доступ до спільних ресурсів [2]:

1. Grid-система координує різні ресурси. Не існує спільного центру управління ресурсами, і система Grid координує їх використання, наприклад, балансування навантаження. Отже, проста система управління ресурсами кластера не є сітковою системою, оскільки вона централізовано управляє всіма вузлами кластера і має повний доступ до них. Сіткова система може мати лише

обмежений доступ до ресурсів, що залежить від політики домену управління (організації власника), де розташовані ресурси.

2. Система мереж базується на стандартах та відкритих протоколах, послугах та інтерфейсах. Без стандартного протокола неможливо легко та швидко підключити нові ресурси до мережі та розробити нові послуги.

Як правило, Grid-система має такі властивості:

– гнучкість, тобто можливість забезпечити розподілений доступ до потенційно будь-якого типу ресурсів;

– масштабованість: продуктивність системи мережі зростає або значно зменшується із складом;

– гнучка та потужна підсистема безпеки: протистояти атакам зловмисників та забезпечувати конфіденційність;

– здатність контролювати ресурси: застосування місцевої та глобальної політики та квот;

– гарантія якості обслуговування;

– можливість використання кількох ресурсів для координації роботи одночасно.

Хоча сама технологія мережі не залежить від конкретних ресурсів, більшість реалізацій мережевих систем забезпечують роботу, яка використовує такі типи ресурсів:

– обчислювальні ресурси – персональні комп'ютери, кластери;

– зберігання ресурсів-дисків і дискових масивів, систем масового зберігання;

– Інтернет-ресурси;

– програмне забезпечення – будь-яке спеціалізоване програмне забезпечення.

Grid технологія охоплює лише найпоширеніші аспекти, однакові для будь-якої системи, такі як, архітектура, протокол, інтерфейс та сервіс. Застосо-

вуючи дану технологію та наповнюючи її конкретним вмістом, ви можете впровадити Grid-систему, призначену для вирішення певних категорій програм.

Мережеві технології не слід плутати з паралельними обчислювальними технологіями. Звичайно, у певній Grid-системі можна використовувати сучасні технології для організації паралельних обчислень, оскільки сіткову систему можна розглядати як метакомп'ютер з безліччю обчислювальних вузлів.

Архітектура Grid

Архітектура Grid визначає системні компоненти, цілі та функції цих компонент і відображає спосіб взаємодії компонентів між собою. Архітектура мережі - це архітектура взаємодіючих протоколів, служб та інтерфейсів, яка визначає основний механізм підключення користувачів до мережі та спільного використання обчислювальних ресурсів для вирішення різних завдань. Архітектура протоколу Grid розділена на кілька рівнів (рис. 1.1), і компоненти на кожному рівні можуть використовувати функції будь-яких компонентів нижчого рівня.

Рис. 1.1. Рівні архітектури протоколів Grid і їх відповідність рівням архітектури протоколів Інтернет

Загально кажучи, ця архітектура встановлює вимоги до основних компонентів технології (протоколи, послуги, інтерфейси прикладних програм та засоби розробки програмного забезпечення) і не забезпечує набору суворих специфікацій, залишаючи можливість її розвитку в рамках визнаної концепції.

Базовий рівень

Базовий (структурний) рівень (Fabric Layer) описує послуги, які безпосередньо використовують ресурси. Ресурс - одна з основних концепцій сіткової

архітектури. Ресурси можуть бути різноманітними, але, як зазначалося вище, існує кілька основних типів: обчислювальні ресурси; ресурси зберігання даних; інформаційні ресурси, каталоги; мережеві ресурси. Обчислювальні ресурси забезпечують обробну потужність для Grid системи користувача (точніше, завдання користувача). Обчислювальним ресурсом може бути кластер або окрема робоча станція. Для будь-яких архітектур будь-яка обчислювальна система може розглядатися як потенційний обчислювальний ресурс сіткової системи. Необхідною умовою для цього є існування спеціального програмного забезпечення, так званого проміжного програмного забезпечення (програмного забезпечення), яке реалізує стандартний зовнішній інтерфейс з ресурсами і дозволяє забезпечувати ресурси мережевій системі.

Головною особливістю обчислювальних ресурсів є продуктивність праці. В якості протосру для зберігання використовується ресурс пам’яті. Для отримання доступу до ресурсу пам’яті, використовується проміжне програмне забезпечення, котре реалізує уніфіковані інтерфейси управління та передачі даних. Як і обчислювальні ресурси, фізична архітектура ресурсів пам'яті не є важливою для мережевих систем, будь то жорсткий диск робочої станції або система масового зберігання з розміром у сотні терабайт.

Обсяг - головна особливість ресурсів пам'яті. Каталогові та інформаційні ресурси - це особливий тип ресурсів пам'яті. Вони призначені для зберігання та надання метаданих та інформації про інші ресурси Grid системи. Інформаційні ресурси дозволяють систематично зберігати велику кількість інформації про поточний стан сіткової системи та ефективно виконувати пошукові завдання.

Зв’язок між розподіленими ресурсами Grid-систем підтримує мережевий ресурс. Головна його особливість – швидкість передачі даних. Географічно розподілені системи, засновані на цій технології, можуть поєднувати тисячі різних видів ресурсів незалежно від їх географічного розташування.

Рівень зв'язку

Рівень зв’язку (з’єднання) (Connectivity Layer) визначає протокол зв’язку та протокол автентифікації. Протокол зв'язку забезпечує обмін даними між ос-

новними компонентами рівня. Протокол автентифікації базується на протоколі зв'язку та забезпечує механізм шифрування для ідентифікації та перевірки користувачів та ресурсів. Протокол рівня зв'язку повинен забезпечувати надійну передачу та маршрутизацію повідомлень, а також присвоювати імена мережевим об'єктам. Незважаючи на існуючі альтернативи, протокол рівня зв'язку в сітковій системі забезпечує використання лише стеку протоколів TCP / IP, особливо: на рівні мережі - IP та ICMP, транспортному рівні - TCP, UDP та на рівні програми - HTTP, FTP, DNS, RSVP.

З огляду на стрімкий розвиток мережевих технологій, майбутній рівень зв'язку може залежати від інших протоколів. З метою забезпечення надійної передачі повідомлень у мережевій системі слід використовувати рішення, що забезпечують гнучкі методи захисту зв'язку (можливість контролю рівня захисту, обмеження делегування повноважень та підтримка надійних протоколів передачі). В даний час ці рішення базуються на загальновизнаних стандартах безпеки (SSL, TLS) та нових розробках, спочатку розроблених для Інтернету.

Ресурсний рівень

Рівень ресурсів (Resource Layer) побудований на протоколі зв'язку та автентифікації архітектури мережі. Рівень ресурсів реалізує протоколи, що забезпечують такі функції:

– координувати стратегію безпеки використання ресурсів;

– ресурсні процедури активації;

– моніторинг стану ресурсів;

– контроль ресурсів;

– облік за використання ресурсів.

Протокол цього рівня покладається на основні функції рівня для доступу та управління локальними ресурсами. На рівні ресурсів протокол використовує єдиний інтерфейс для взаємодії з ресурсами без розрізнення архітектурних характеристик конкретних ресурсів.

Існує два основних типи протоколів рівня ресурсів:

– отримувати інформаційний протокол про структуру ресурсу та інформацію про стан, наприклад, його конфігурацію, поточне навантаження, стратегію використання;

– протоколи управління, що використовується для узгодження доступу до спільних ресурсів, визначення вимог до ресурсів та ефективних операцій (наприклад, підтримка надмірності, можливості процесів, доступ до даних).

Угода про управління повинна перевірити, чи відповідає запитувана операція політиці розподілу ресурсів, включаючи бухгалтерський облік та можливі платежі. Вони можуть підтримувати функції моніторингу стану та управління транзакціями. Список протоколів протоколу на рівні ресурсу наближається до списка основних рівнів архітектури Grid. Додана лише вимога підтримувати єдину семантику різних операцій системи сповіщення про помилки.

Колективний рівень

Колективний рівень (Collective Layer) відповідає за глобальну інтеграцію різних наборів ресурсів, а не рівень ресурсів, який зосереджений на використанні одного ресурсу. На колективному рівні існують загальні та конкретні (специфічні для заявки) угоди. Загальні протоколи спочатку включають протоколи ідентифікації та розподілу ресурсів, системи моніторингу та авторизації громади. Для різних мережевих додатків (наприклад, протокол розподіленого архівування даних або протокол управління державними завданнями тощо) були створені конкретні протоколи.

Компоненти колективного рівня забезпечують широкий спектр технологій розподілу ресурсів. Нижче наведені функції та послуги, реалізовані на цьому рівні угоди:

– довідник служби дозволяє віртуальним організаціям знаходити безкоштовні ресурси, запитувати імена та атрибути ресурсів, такі як тип і завантаження;

– послуги спільного розподілу, планування та розподілу ресурсів забезпечують розподіл одного або декількох ресурсів з певною метою, а також планування завдань, що виконуються з використанням цих ресурсів;

– служби моніторингу та діагностики відстежують аварії, напади та перевантаження;

– послуги реплікації даних координують використання ресурсів пам'яті у віртуальних організаціях та забезпечують більший доступ до даних на основі вибраних показників (таких як час відгуку, надійність, вартість тощо);

– послуга управління навантаженням використовується для опису та управління багатоступеневими, асинхронними, багатокомпонентними завданнями;

– служби авторизації громади допомагають вдосконалити правила доступу до розподілених ресурсів та визначити можливості використання ресурсів громади. Цей тип послуг дозволяє розробляти політику доступу на основі інформації про ресурси, протоколів управління ресурсами та протоколів безпеки на рівні каналів;

– бухгалтерські та платіжні служби забезпечують збір інформації про використання ресурсів для контролю запитів користувачів;

– координаційні служби підтримують обмін інформацією між потенційно великими спільнотами користувачів.

Прикладний рівень

Прикладний рівень (Рівень додатків) (Application Layer) описує спеціальний додаток, що працює у віртуальному організаційному середовищі. Додаток працює з використанням служб, визначених на наступних рівнях. На кожному рівні існують певні протоколи, що забезпечують доступ до необхідних служб, а також інтерфейси прикладного програмування (API), відповідні цим протоколам.

Щоб було легше використовувати інтерфейс прикладного програмного забезпечення, ми надаємо користувачам комплект для розробки програмного забезпечення (SDK). Набір інструментів високого рівня може забезпечити можливість одночасного використання декількох протоколів та поєднання операцій

протоколу з додатковими викликами інтерфейсів прикладного програмного забезпечення низького рівня.

Слід зазначити, що справжній додаток можна викликати через досить складні оболонки та бібліотеки. Самі ці оболонки можуть визначати протоколи, служби та інтерфейси прикладного програмування, але ці додаткові компоненти не є основними протоколами та послугами, необхідними для побудови системи Grid.

Віртуальна організація

Інфраструктура мережі Grid базується на спільних ресурсах, з одного боку, та використанні загальнодоступних ресурсів, з іншого. У зв'язку з цим ключовою концепцією мережевої інфраструктури є віртуальна організація споживачів та власників ресурсів, що працюють разом. Мотивація співпраці може бути різною. У існуючій сітковій системі віртуальна організація – це об’єднання (співпраця) експертів з певної галузі застосування, які об’єднуються для досягнення спільної мети. Будь-яка віртуальна організація має певну кількість ресурсів, що надаються її зареєстрованим власником (деякі ресурси можуть належати декільком віртуальним організаціям одночасно).

Кожна віртуальна організація встановлює власні правила роботи для своїх членів на основі балансу між потребами користувачів та грошовими ресурсами, тому користувачі повинні довести своє бажання співпрацювати з мережевою системою та отримати згоду агентства з управління віртуальною організацією. Сіткова система (Grid-система) – це колективне обчислювальне середовище, в якому кожен ресурс має власника, а доступ до ресурса відкривається в режимі, розподіленому в часі та просторі, і багато користувачів є частиною віртуальної організації. Віртуальні організації можуть створюватися динамічно та мати обмежений життєвий цикл.

Отже, мережеву систему можна визначити як просторово розподілене робоче середовище з гнучким, безпечним та скоординованим розподілом ресурсів для запуска програм у певних віртуальних організаціях. Поки що існує багато віртуальних організацій, які є частиною різних мережевих систем.

GRID - це інструмент для обміну обчислювальними потужностями та файлами даних через Інтернет. Це дозволяє вийти за межі простого обміну даними між комп'ютерами і перетворити глобальну комп'ютерну мережу на майже необмежений обчислювальний ресурс.

Прикладами віртуальних організацій, що працюють за проектом LCG-2 (сітка обробки даних прискорювача CERN LHC), є віртуальні організації для експериментів, запланованих на прискорювачі: ATLAS, CMS, Alice, LHCb.

Використання інфраструктури GRID у наукових цілях

У більшості випадків прикладом такої інфраструктури є найбільша у світі мережева інфраструктура, реалізована в рамках проекту EGEE (Enable Grid ESCIENCE). Метою проекту EGEE є інтеграція національних, регіональних та тематичних розробок мереж, які були розпочаті, в єдину інтегровану мережеву інфраструктуру для підтримки досліджень. Ця інфраструктура забезпечує дослідникам наукових кіл та різних економічних галузей цілодобовий доступ до високопродуктивних обчислювальних ресурсів, незалежно від їх географічного розташування.

Інфраструктура буде надана спільноті географічно розподілених дослідників, які потребують загальних обчислювальних ресурсів і готові об'єднати власну обчислювальну інфраструктуру та погодитись на принцип спільного доступу. Проект в основному підтримується фінансовими агентствами ЄС, але він розроблений для науковців усього світу. Значний обсяг фінансування надходить від США, Росії та інших учасників проекту, що не входять до ЄС.

Чотирирічний план проекту EGEE складається з двох етапів. Перший етап (з 1 квітня 2004 по 31 березня 2006 рр.) закінчився. ругий етап проекту (EGEE-II) розпочався 1 квітня 2006 року. Проект стартував у надзвичайно сприятливих умовах: до офіційного запуску були випущені основні служби, розпочато розробку проміжного програмного забезпечення та розповсюдження інформації. Було обрано два практичні напрямки для визначення початкового рівня впровадження мережевої інфраструктури, що розвивається, та офіційної оцінки її експлуатаційної якості та функціональності. Одним із них є обробка

даних експериментів з прискорювачами LHC, де інфраструктура мережі забезпечує зберігання та аналіз великої кількості реальних та змодельованих даних фізичних експериментів високих енергій, проведених у ЦЕРНі. Інший – це біомедична сітка, в якій багаторазові колаборації вирішували однаково складні проблеми, такі як пошук генома бази даних та індексація бази даних лікарні, що становить кілька терабайт даних на рік для лікарні.

На сьогоднішній день багато програм використовують цю інфраструктуру, що розвивається, у різних галузях науки: термоядерний синтез, наука про Землю, астрофізика, геофізика, археологія та обчислювальна фізика. Інфраструктура також відкрита для промислових та соціально-економічних спільнот. У цій інфраструктурі було створено понад 60 віртуальних організацій. Понад 90 організацій з 32 країн взяли участь у проекті EGEE. Ці організації об'єднані в регіональну сітку (конфедерацію).

Проект EGEE встановив партнерські відносини з понад 70 учасниками, що не належать до проекта, в тому числі за допомогою багатьох інших проектів Grid. Інфраструктура EGEE базується на дослідницькій мережі GEANT Європейського Союзу, яка надає можливості для співпраці з іншими мережевими системами у світі, включаючи США та Азію, що сприяє формуванню глобальної мережевої інфраструктури. З 2007 року була створена Європейська мережева інфраструктура (EGI), і планується постійно діяти як Національна координаційна мережа Grid Infrastructure (NGI). ЦЕРН залишає за собою загальну відповідальність та відповідальність за оновлення проміжного програмного забезпечення та загальної системи безпеки. Росія та Україна майже одночасно почали співпрацювати з ЦЕРН з метою побудови національної мережевої інфраструктури. У Росії Російський альянс інтенсивних мереж даних (RDIG) був створений в 2003 році для забезпечення ефективного розповсюдження інфраструктури EGEE в Росії, а також інших організацій у різних галузях науки, освіти та промисловості. До її складу входять 11 провідних фізичних інститутів, 4 університети та Геофізичний центр Російської академії наук. Офіційна презентація першого в Україні сегменту електромереж відбулася в Інституті

теоретичної фізики на засіданні Президії Національної академії наук України 4 квітня 2007 року.

В даний час ця частина поєднує в собі обчислювальні ресурси такої організації: Інститут теоретичної фізики. М. М. Боголюбова НАН України, Інститут фізики конденсованих речовин НАН України (Львів), Інститут віртуальних машин імені кібернетики Україна Глушкова НАН України, НАН України та Космічний інститут НДА, Головна астрономічна обсерваторія НАН України, НАН України Молекулярна Інститут біології науки та генетики, Україна Інститут клітинної біології та генної інженерії ім. На зустрічі було обговорено не лише розвиток мережевих технологій в Академії наук, але також розглянуто та окреслено перспективи співпраці між Національною академією наук України та Міністерством освіти і науки (МОН) України, особливо з Національним Технічний університет України «Київський політехнічний інститут». Створення Національної сітки України (UAG). Зокрема, експерти НТУ «КПІ» брали участь у проекті обробки даних Великого адронного колайдера ЦЕРНу.

Разом із експертами Інституту теоретичної фізики імені М.М. Боголюбова НАН України вчені КПІ брали участь у європейському проекті EGI (Європейська мережева ініціатива). Для виконання цього завдання для відповідних обчислень було виділено 200 обчислювальних ядер університетського кластера. НТУУ «КПІ» брав участь у розробці Міжнародної мережі освітніх мереж спільно з науковцями Російської Федерації, Болгарії та Казахстану. Незважаючи на брак обчислювальних ресурсів, розпочато проект Міністерства освіти і науки України та Національної академії наук «Розробка та впровадження сервісів семантичної мережі для інтелектуальної обробки даних». Кілька наукових та освітніх установ в Україні висловили зацікавленість взяти участь у проекті як партнери.

Сотні мережевих систем представляють різні аспекти технології, реалізовані на її основі проекти та приклади різних застосувань. Все це не тільки показує масштаб, але і відображає результат впровадження нового методу організації обчислювальних процесів у різних галузях науки, техніки та економіки.

Поточними основними завданнями українського енергетичного сектору є: розширення інфраструктури, збільшення потужності ресурсного центру, пошук завдань, які можна вирішити за допомогою мережевих технологій, навчання користувачів та залучення молодих спеціалістів для розробки та використання системи та додатків рівень технології в XXI ст.

1.2. Аналіз існуючих вразливостей

Поняття «вразливість» описане в таких стандартах, як ISO / IEC 29147: 2014 Information technology. Security techniques. Vulnerability disclosure (Україні існує стандарт ДСТУ ISO / IEC 29147: 2016 Інформаційні технології. Методи захисту. Розкриття вразливості) та ISO/IEC 27000:2016. ISO / IEC 29147 стверджує, що вразливі місця – це вразливості програмного, апаратного забезпечення або онлайнових служб. Слабкість системи може бути спричинена дефектами дизайну програмного та апаратного забезпечення, дефектами управління процесом розробки тощо [5].

ISO / IEC 27000: 2016 визначає вразливості як слабкі сторони активів або інструментів контролю та управління, які можуть бути використані однією або кількома загрозами [6]. У документі ND TZI 1.1-003-99 вказується на вразливість системи - система не може протистояти реалізації конкретної загрози або групи загроз [7].

Список CWE (Common Weakness Enumeration), складений MITER, описує помилки, які можуть виникати на різних етапах життєвого циклу програмного забезпечення. Згідно з веб-сайтом MITER, CWE є основним стандартом для виявлення та запобігання вразливості програмного забезпечення [10]. Кожному дефекту у списку присвоюється власний ідентифікатор (ID), і список містить приблизно тисячу ідентифікаторів CWE. При описі вразливості в базі даних вразливості може бути вказаний ідентифікатор CWE для отримання додаткової інформації.

Кожен ідентифікатор CWE у списку CWE має такий опис [11]:

– короткий та/або розширений опис конкретного ідентифікатора CWE;

– помилки, які можуть зробити розробники, можуть призвести до лазівки в майбутньому (фаза проектування, експлуатація);

– мова/платформа програмування, яка є основою програмного забезпечення (Java, C ++ або інше);

– як приклад, користувач також може навести неправильні фрагменти програмного коду, що може призвести до несправностей програмного забезпечення або вразливостей;

– посилання на записи в міжнародній базі даних про вразливості, що містять дані про існуючі вразливості в компонентах програми, спричинені помилками з ідентифікаторами CWE;

– варіанти корегування неправильного способу або введення правильної версії програмного коду, щоб забезпечити безпеку обробки даних;

– Посилання на інші ідентифікатори CWE, які спричинили помилки в певному місці програмного забезпечення на конкретному етапі.

Іншим варіантом класифікації вразливостей та ризиків є рейтинг 10 найбільш поширених вразливостей та ризиків - OWASP Top-10. OWASP - це некомерційна організація, діяльність якої зосереджена на поліпшенні програмного забезпечення [12]. Остання версія документа OWASP Top-10 була випущена OWASP у 2017 році, і його зміст є таким [13]:

– вставка інструкцій (Injection) – це вставка SQL - операторів, що надсилаються на обробку, які після їх отримання можуть розпочати виконання будь-яких команд;

– некоректна (порушена) автентифікація (Broken Authentication) – функції автентифікації можуть бути реалізовані таким чином, що дозволяє обійти паролі або отримати ідентифікатори користувачів;

– витік критичних даних (Sensitive Data Exposure) – вплив на конфіденційні дані – недостатній захист персональних даних;

– атаки на засоби аналізу XML External Entities – зловмисники можуть завантажувати XML-файли на сервер або розміщувати шкідливий код у XML-документі;

– неправильний контроль доступу (Broken Access Control) – контроль доступу реалізований таким чином, що авторизовані користувачі можуть мати привілеї в системі, яких вони не повинні мати;

– небезпечна конфігурація безпеки (Security Misconfiguration) – відсутність оновлень та неправильна конфігурація окремих компонентів може становити додатковий ризик для безпеки;

– міжсайтове виконання сценаріїв (XSS) – зловмисник отримує можливість запускати сценарії в браузері жертви, перехоплювати скрипти користувачів, перенаправляти користувачів на інші веб-сайти;

– незахищена десеріалізація (Insecure Deserialization) – зловмисник може порушити логіку програми, підробляючи об'єкти програми, що призводить до віддаленого виконання коду зловмисника;

– використання компонентів з відомими вразливостями/місцями (Using Components with Known Vulnerabilities) – програмне забезпечення, термін дії якого не закінчився або не оновився, може залучити більше зловмисників до свого злому;

– недостатня реєстрація та моніторинг (Insufficient Logging and Monitoring) – погано організований механізм реєстрації та моніторингу подій може змусити зловмисників неодноразово атакувати непомітно.

Згідно з дослідженням компанії Vercode, що займається тестуванням безпеки програмного забезпечення, 74% програмних систем мають принаймні одну вразливість у тесті на вразливість OWASP Top-10 [14]. Враховуючи ці результати тестування, необхідно проаналізувати наявну інформацію про вразливості зі списку OWASP Top-10, щоб підвищити ефективність їх виявлення під час тестування на проникнення. Нижче наведено більш детальний аналіз вразливості зі списку 10 найкращих OWASP.

Вставка інструкцій

Прикладом вставки інструкцій може бути вставка SQL – інструкцій. Вставка операторів SQL, залежно від типу використовуваної системи управління базами даних (СУБД), може дозволити зловмиснику виконати будь-який запит до бази даних (прочитати вміст таблиці, видалити, змінити або додати дані), прочитати та/або записати локально файли та виконувати будь-які команди на цільовому сервері зловмисником. Атака «вставити SQL» може бути реалізована через неправильну обробку вхідних даних, що використовуються в запитах SQL. Потім аналіз прикладу вставки SQL-атаки - інструкції [15].

Якщо є серверне програмне забезпечення, яке отримує вхідний параметр id і використовує його для побудови запиту SQL, PHP - сценарій обробки запитів може виглядати так: $id = $_REQUEST['id']; $res = mysqli_query («SELECT * FROM news WHERE id_news = » . $id); Якщо серверу передано параметр id, рівний 5 (http://example.org/script.php?id=5), тоді буде зроблений запит SQL: SELECT * FROM news WHERE id_news = 5. Але якщо зловмисник передасть в якості параметра id рядок -1 OR 1 = 1 (http://example.org/script.php?id=-1+OR+1=1), то буде зроблено запит: SELECT * FROM news WHERE id_news = -1 OR 1 = 122

Отже, зміна вхідних параметрів шляхом додавання до них конструкцій мови SQL змінює логіку виконання запитів SQL (у наведеному вище прикладі замість запису із заданим ідентифікатором будуть вибрані всі доступні записи в базі даних) [15].

Некоректна автентифікація

Згідно з дослідженнями, зловмисники можуть мати доступ до сотень мільйонів дійсних комбінацій імен користувачів та паролів для заповнення форм, списків адміністрування облікових записів та інструментів для пошуку значень у словнику [17]. Зловмисникові досить отримати доступ лише до кількох облікових записів або лише до одного облікового запису адміністратора, щоб проникнути в систему, викрасти гроші, скористатися оманливими методами, викрасти конфіденційну інформацію тощо.

Атаки автентифікації можливі, якщо система має такі вразливості [17]:

– місця для заповнення облікових даних, коли зловмисник має список дійсних імен користувачів та паролів;

– відсутній захист від виконання атак методом пошуку;

– дозволені прості або звичні паролі, такі як «пароль» або «адміністратор/адміністратор»;

– існують неефективні процеси відновлення облікових даних та забутих паролів, які неможливо захистити;

– відсутність багатофакторної автентифікації;

– надання ідентифікаторів сеанссеансів в URL – адресі.

Прикладом атаки на засоби автентифікації є випадки, коли зловмисник знає секретний ідентифікатор сеансу та може представитись веб-серверу авторизованим користувачем та зламати його обліковий запис. Якщо система недостатньо захищає ідентифікатори сеансу (відображаючи ідентифікатори всередині URL-адреси замість використання файлів cookie), зловмисник може дуже легко отримати ідентифікатор сеансу. Навіть якщо система зберігає ідентифікатори сеансів у файлах cookie, зловмисник все одно може отримати необхідну інформацію з локальних файлів користувача, змушуючи його запустити маскований сценарій. Простий скрипт можна використовувати для пошуку ідентифікатора сеансу, який зберігається у файлі cookie - файлі на комп'ютері користувача. Прикладом є отримання ідентифікатора сеансу за допомогою сценарію, введеного в поле вразливого пошуку системи.

З метою підвищення рівня безпеки системи автентифікації та управління сеансами в системі необхідно використовувати сучасні платформи для розробки програмного забезпечення (програмного забезпечення) та техніки написання захищених кодів. Особливо при зміні важливих налаштувань облікового запису (наприклад, паролів або електронних адрес), ви можете прив’язати вихідну IP-адресу до ідентифікатора користувача сеансу та примусити повторну автентифікацію. Після злому мережевого сеансу зловмисник негайно спробує це зробити. У цих випадках такі прості методи програмування, як повторна автентифі-

кація, можуть порушити несанкціоновані сеанси та зменшити ризик хакерських атак [16].

Ще одним простим прийомом програмування, який не можна ігнорувати, є видалення сеансів після заздалегідь визначеного простою. Наприклад, якщо користувач не працює з сайтом протягом п’яти хвилин, система припиняє сеанс, змушуючи користувача (або будь-якого зловмисника) повторно пройти автентифікацію, щоб продовжувати працювати. Згідно з дослідженнями, цей метод також зменшує ризик використання несанкціонованих сесій і застосовується у більшості банківських систем [16].

Витік критичних даних

Згідно з дослідженнями, крадіжка критичних даних була найпоширенішою атакою за останні кілька років. Основним недоліком є відсутність шифрування конфіденційних даних або коли криптографічні алгоритми, генерація та управління ключами нестабільні [18]. Для того, щоб визначити можливість критичного витоку даних, необхідно визначити, які дані потребують захисту в кожному випадку під час їх передачі та зберігання. Така інформація може включати паролі, номери кредитних карток, медичні записи, особисту інформацію та ділові записи або іншу конфіденційну інформацію. Для всіх цих типів даних перевірте [18]:

– чи є дані, що надсилаються у звичайному тексті (це стосується таких протоколів, як HTTP, SMTP та FTP);

– чи використовуються старі або нестабільні хакерські криптографічні алгоритми в системі або в програмному коді;

– як реалізований механізм розподілу та управління криптографічним ключем;

– чи використовує поштовий клієнт недійсний сертифікат, отриманий від сервера.

Одним з сценаріїв витоку критичних даних може бути процес шифрації програмою номерів кредитних карток у базі даних за допомогою автоматичного шифрування бази даних. Проте ці дані автоматично розшифровуються при за-

вантаженні, дозволяючи застосовувати вставки SQL – інструкцій для отримання номерів кредитних карт у вигляді відкритого тексту. Іншим сценарієм витоку критичних даних може бути веб – сайт, у якому не використовується протокол захисту транспортного рівня TLS (Transport Layer Security) для всіх сторінок або використовуються нестійкі до зламу криптографічні алгоритми шифруван-ня. Зловмисник може здійснити моніторинг мережевого трафіку (наприклад, в незахищеній бездротовій мережі), знизити рівень зв'язків з HTTPS до рівня HTTP, перехопити запити та викрасти дані з cookie користувача. Потім зловмисник відтворює цей файл cookie та викрадає сеанс користувача (аутентифікований), отримуючи доступ або змінюючи особисті дані користувача. Більше того, одержувач переказу може змінитися таким чином. Крім того, використання простих алгоритмів для обчислення хешу пароля може призвести до витоку критичних даних. У цьому випадку всі хеші можна обчислити, використовуючи райдужні таблиці попередніх хешів, використовуючи графічні процесори [18].

Атаки на засоби аналізу XML – вводу

XML (eXtensible Markup Language) – є стандартом для обміну структурованими даними у текстовому форматі. Вміст XML-файлу може виглядати так [19]: version=«1.0» encoding=«UTF-8»?> 27 1234 1 Jan P. Monsch 789 Можливі цілі:

– веб-сервіс;

– генератор XML;

– синтаксичний аналізатор XML;

– програмний код.

На рис. 1.2 показано можливі місця нападу на засоби аналізу вхідних даних XML [19].

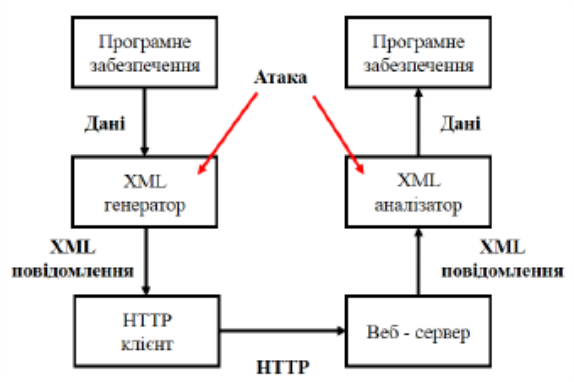

Рис. 1.2. Можливі місця здійснення атаки на засоби аналізу XML – вводу

Можливі приклади організації атак на засоби аналізу вхідних даних XML представлені нижче [20]. На рис. 1.3 показаний приклад визначення типу даних з іменем foo, за допомогою елемента bar, асоційованого зі словом «World». Тому аналізатор XML у будь-який час замінить цей об’єкт на «World».

Request

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

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

Response

```
HTTP/1.0 200 OK

Hello World
```

Рис. 1.3. Приклад передачі XML – елементів до аналізатора

Згідно з дослідженнями, об'єкти XML можуть використовуватися зловмисниками для запуску атак «Відмова в обслуговуванні», вставляючи об'єкти з елементами, що містять інші елементи всередині тощо [20]. На рис. 1.4 наведено приклад запиту до синтаксичного аналізатора XML та результат обробки запиту.

Request

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

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

Response

```
HTTP/1.0 200 OK

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

Рис. 1.4. Приклад передачі до XML – аналізатора обєкта з вкладеними елементами та результат обробки запиту

Відправляючи необхідні запити до синтаксичного аналізатора XML, зловмисник, який має інформацію про структуру системи, може отримати інформацію про використовуване програмне забезпечення [20]. Приклад можливих інструкцій у XML-файлі та результат аналізатора XML показані на рис. 1.5.

Рис. 1.5. Приклад передачі до XML – аналізатора об'єкта з інструкціями на визначення операційної системи

Неправильний контроль доступу

Метою контролю доступу є забезпечення дотримання вимог політики безпеки таким чином, що користувачі системи не можуть виконувати дії, що перевищують їх дозволи. Прикладом контролю доступу є обмеження доступу до ресурсів неавторизованих користувачів [21]. Однак, як відомо, обмеження доступу до ресурсів можуть поширюватися не лише на файли та бази даних, а й [22]:

– програми;

– спеціальні функції системи;

– спеціальні поля даних;

– пам'ять;

– приватні або безпечні змінні;

– носії інформації;

– засоби передачі інформації.

Недоліки в засобах контролю доступу зазвичай призводять до несанкціонованого розкриття, модифікації або знищення всіх даних або виконан-

ня ділових функцій за межами користувача. Загальні уразливості контролю доступу включають [21]:

– обхід контролю доступу, змінивши URL - внутрішній стан програми, HTML - сторінку або просто за допомогою спеціального інструменту для атаки на програмний інтерфейс;

– можливість використовувати первинний ключ для зміни всіх записів інших користувачів, що дозволяє переглядати або редагувати облікові записи інших людей;

– збільшення прав (прав користувача в системі без проходження етапів авторизації або прав адміністратора після входу в систему як користувач);

– маніпулювання метаданими, таке як повторне використання або заміна маркерів контролю доступу або файлів cookie, які можуть бути використані для збільшення системних дозволів.

Небезпечна конфігурація оточення

Згідно з дослідженнями OWASP, можливість неправильної конфігурації середовища, тобто операційного середовища системи, може виникати на будь-якому рівні системних елементів, включаючи мережеві служби, платформу, веб-сервер, сервер, базу даних попередньо встановлених віртуальних машин тощо. Автоматизовані сканери вразливості можуть виявляти недійсні конфігурації, використовувати облікові записи або стандартні конфігурації тощо. Наявність автоматизованих засобів виявлення вразливості допомагає зловмисникам отримати несанкціонований доступ до певних даних або системних функцій. Час від часу, такі недоліки призводять до повного розкриття структури системи та викрадення конфіденційних даних тощо [23]. Додатковим ризиком є наявність програмного забезпечення, яке не оновлюється. Можливі сценарії атаки через небезпечну конфігурацію:

– програмне забезпечення, встановлене на сервері, не було видалено після завершення проектування або тестування системи (програмне забезпечення містить відомі уразливості і може використовуватися зловмисниками), а тестовий обліковий запис не змінено;

– Конфігурація сервера дозволяє отримувати докладні повідомлення про помилки (інформацію про використовуване програмне забезпечення, версію та вразливості у зазначеному програмному забезпеченні, яке зловмисник може шукати).

Міжсайтове виконання сценаріїв

Атака на веб-систему, яка вразлива для атак міжсайтових сценаріїв, полягає в тому, щоб вставити код зловмисника на сторінку веб-системи, яка буде виконана на комп'ютері користувача (коли він відкриє сторінку), і код взаємодіє з комп'ютером зловмисника. Мережевий сервер. Цей тип атаки називається XSS-атакою [24]. Acunetix - компанія, яка розробляє рішення для поліпшення безпеки веб-додатків, на веб-сайті якого визначено схеми атак XSS, як показано на рис. 1.6 [24].

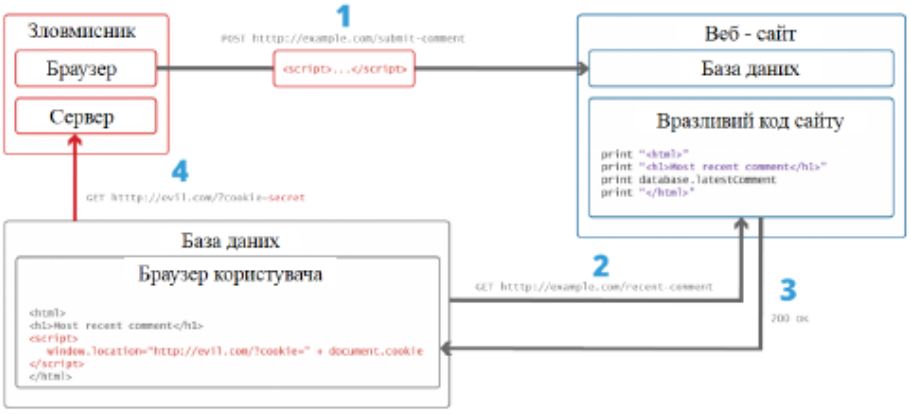

Рис. 1.6. Схема здійснення XSS – атаки

На рис. 1.6 цифрами позначено наступні дії:

1 – зловмисник знаходить вразливий веб-сайт і вставляє скрипт у форму введення даних;

2 – користувач генерує запит на веб-сторінку з веб-сайту; веб-сервер надсилає користувачеві сторінку з кодом зловмисника як частину HTML-документа;

3 – користувацький браузер виконує код зловмисника в документі HTML і відправляє файл cookie користувача на сервер зловмисника, після чого (дія 4);

4 – зловмисник може використовувати вкрадені файли cookie - файли жертви, щоб рекламувати себе як іншу особу.

Для того, щоб код зловмисника запускався на сторінці користувача, зловмисник може використовувати технології соціальної інженерії, щоб переконати користувача відкрити сторінку зі сценарієм завантаження. З огляду на можливі збитки, спричинені такими атаками, XSS-атаки включені до списку 10 найкращих OWASP і є однією з найпоширеніших атак.

Незахищена десеріалізація

Процес серіалізації використовується для передачі об’єктів по мережі та збереження їх у файлах. Для цього заповніть об’єкт необхідними даними, потім викличте код серіалізації, і в результаті вийде документ XML або інший відформатований документ. Серіалізований результат передається одержувачу електронною поштою або HTTP. Приймач створює об'єкт того ж типу, що і до серіалізації, тим самим виконуючи процес десериалізації. Згідно з дослідженнями Acunetix, більшість загроз інформаційній безпеці виникає під час процесу десериалізації. На даний момент надсилається не серіалізований об’єкт, а подібний об’єкт із кодом зловмисника.

Успішна десеріалізація обєкта зловмисника може призвести до таких атак, як «відмова в обслуговуванні», заміна даних користувача під час автентифікації та віддалене виконання коду [25]. Відповідно до OWASP, серіалізацію можна використовувати для наступних елементів веб-системи [26]:

– елементи розподіленої системи;

– мережевий протоколи, мережеві сервіси;

– інстумент кешу;

– база даних, файлова системи;

– HTTP – файли cookie, параметри форми HTML, маркери автентифікації.

Використання компонентів з відомими вразливостями

У структурі мережевої системи використовується безліч компонентів, і їх виробники можуть бути різними. По-перше, використовуючи різні компоненти під час побудови веб-системи, майте на увазі, що виробник може допускати помилки при створенні компонентів на всіх можливих етапах, що призводить до уразливості цих компонентів. Компонентами веб-системи можуть бути ре-

алізація різних служб, сторонні бібліотеки, системи управління вмістом, реалізація веб-серверів, операційних систем тощо. По-друге, якщо є вразливості компонентів, виробники компонентів, користувачі або зловмисники можуть це виявити. Завданням виробника є виправлення вразливості. Однак для усунення вразливості може знадобитися деякий час, і зловмисник може використати цей час для нападу на користувача програми, що містить вразливий компонент. Слід зазначити, що у випадку виробників або користувачів інформація про вразливість може бути розкрита загальнодоступному після оновлення компонента (користувач виявляє вразливість та інформує виробника). Однак, якщо користувач виявляє вразливість у компоненті та використовує цю інформацію для використання вразливості, його або її можна вважати зловмисником. Якщо ви зловмисник, ви можете отримати інформацію про вразливість на спеціальному форумі зловмисника.

Слід також додати, що багато користувачів не встановлюватимуть оновлення програмних компонентів, які вони використовують, піддаючи систему та власні дані ризику. З огляду на доступність інформації про вразливості, техніки пошуку та використання вразливостей та програмного забезпечення для уразливості у формі відкритої бази даних про вразливості, обов’язковою є перевірка компонентів, які можуть містити відомі вразливості.

Недостатній моніторинг та реєстрація інцидентів. Як ми всі знаємо, злочинці можуть використовувати процес моніторингу та реєстрації для поганої роботи - це основа майже всіх великих інцидентів. Однією зі стратегій, щоб визначити, чи впроваджено достатній рівень моніторингу, є перевірка журналу подій після тестування на проникнення. Поведінка тестувальника повинна бути записана таким чином, щоб зрозуміти, яку шкоду може спричинити поведінка потенційного зловмисника. Згідно з дослідженнями, у 2016 році час виявлення вторгнення становив 191 день - достатньо часу для заподіяння шкоди [7].

Прояви реалізації недостатнього моніторингу та ведення журналів подій відбуваються, коли [7]:

– спроба ввести ім'я для входу, пароль та помилка транзакції не реєструється;

– метод реєстрації події не передбачає фіксації підозрілої діяльності;

– журнали подій зберігаються лише локально;

– під час випробування на проникнення не надходило повідомлення про вторгнення;

– повідомлення про помилки записуються в журнал подій, але характеристики події чітко не вказані.

Відповідно до [1], рішення про те, як забезпечити захист інформації в комп'ютерній системі (КС), де застосовувати ці механізми та якими вони повинні бути, вважається результатом всебічного аналізу. Аналіз повинен включати виявлення інформаційних загроз, оцінку ймовірності таких загроз та величини потенційних збитків, а також оцінку ризиків, пов'язаних із реалізацією цих загроз, як функції ймовірності та величини потенційних втрат. Існує багато різних методів оцінки ризику. Тому для розрахунку ризику в [9] пропонується наступна формула:

$R = \sum_{j=1}^{N} R_j g_j,$

де $\sum_j g_j = 1$, R – узагальнений показник ($0 \leq R \leq 1$), R_j – кількісна оцінка j-го виду ризику ($0 \leq R_j \leq 1$), g_j – його вага.

У свою чергу, кількісна оцінка j-го виду ризику становить:

$R_j = \frac{1}{m} \sum_{i=1}^{n_j} R_{ij} g_{ij},$

де $\sum_j g_{ij} = 1$, R_{ij} – бальна оцінка і-го фактора в j-у виді ризику, g_{ij} – вага і-го фактора в j-у виді ризику, n_j – кількість врахованих факторів у j-у виді ризику, m – розмах бальної шкали, в межах якої здійснюється оцінка факторів. Методика оцінювання базового та залишкового ризику. Найпоширенішою та використовуваною за відсутності СЗІ є формула базового ризику (за умов, що загрози складають повну групу подій, [10]):

$R = \sum_{i=1}^{N} P_i Q_i,$

де P_i – ймовірність реалізації і-ї загрози, Q_i – втрати, що можуть виникнути за умови реалізації і-ї загрози. Сама ймовірність реалізації і-ї загрози становить добуток:

$$P_j = P_{\partial_j} P_{\partial\phi_j},$$

де P_{∂_j} – ймовірність появи джерела створення сприятливих умов (середовища) для проявлення j-го фактора, $P_{\partial\phi_j}$ – ймовірність проявлення j-го дестабілізуючого фактора. Щодо вразливості системи, то згідно [10] її характеризує залишковий ризик:

$$R_o = \sum_{i=1}^{N} p_i P_i Q_i,$$

де R_o – залишковий ризик, p_i – ймовірність реалізації і-ї загрози після встановлення СЗІ, тобто залишкова ймовірність, оскільки нульової досягнути неможливо.

Отримана таким чином оцінка залишкового ризику може теоретично визначити найнебезпечніші інформаційні загрози в системі, завдяки чому можуть бути використані деякі конкретні механізми захисту. Використовуйте експертну оцінку для визначення ризику на основі двох факторів. В реальній системі кількісна оцінка вищезазначених значень може бути дуже складною, іноді навіть неможливою. Наприклад, важко безпосередньо розрахувати величину збитків, які можуть бути спричинені реалізацією різних загроз, оскільки оцінка тієї чи іншої інформації невідома. Потім, щоб вирішити проблему, вони звернулися до теорії прийняття рішень та методів експертної оцінки. Так, в [11] розглянуто показник ефективності реалізації загрози інформації E_{ij}, який визначено як показник оцінки ризику, пов'язаного з реалізацією даної загрози:

$$E_{ij} = Q_{ij} R_{ij},$$

де Q_{ij} – показник, що характеризує відносний внесок і-ї загрози інформації, реалізованій на протоколі j-го рівня КС, до сумарного ефекту дій зловмисника; R_{ij} – показник, що характеризує статистичну ймовірність реалізації і-ї загрози інформації, реалізованій на протоколі j-го рівня КС.

Значення показника Q_{ij} можна отримати, використовуючи Процес аналітичної ієрархії (MAI) [12], і ієрархію пріоритетів характеристик інформаційної загрози слід будувати таким чином:

- 1-й рівень – пріоритет цілі (порушення конфіденційності, цілісності або доступності);
- 2-й рівень – пріоритет інформаційного об'єкта, реалізованого загрозою;
- 3-й рівень – пріоритет компонентів КС, реалізованих загрозою;
- 4-й рівень – пріоритет рівня стека протоколів, який реалізує загрозу.

Саме значення показника Q_{ij} для і-ї загрози інформації n-го типу, реалізованої щодо k-го інформаційного об’єкта в m-у компоненті КС на протоколі j-го рівня, визначається за формулою:

$$Q_{ij} = T_n O_{kn} C_{mk} P_{jm},$$

де T_n – елемент вектора приоритетів типів загроз $\{T_n\}$, що відповідає n-у типу загрози; O_k – елемент відповідаючого n-у типу загрози вектора пріоритетів $\{O_{kn}\}$, що відповідає інформаційному об’єкту k-го типу; C_{mk} – елемент відповідаючого інформаційному об’єкту k-го типу вектора пріоритетів $\{C_{mk}\}$, що відповідає компоненту КС m-го типу; P_{jm} – елемент відповідаючого компонента КС m-го типу вектора пріоритетів $\{P_{jm}\}$, що відповідає j–му рівню стеку протоколів. Значення показника R_{ij} можна знайти за формулою:

$$R_{ij}(j) = aj^3,$$

Де $a = \frac{1}{\sum_{k=1}^{N} k^3}$ – коефіцієнт нормування, N – кількість рівнів стека протоколів (для стека TCP/IP – 4), j – номер рівня стека протоколів.

Цей метод має характеристики негативного впливу на рішення обрати його в якості методу аналізу ризику вразливостей інформаційної безпеки мережі. Ця стаття бере цей метод як приклад. По-перше, метод розглядає лише оцінку ризику загроз за двома факторами. Цими факторами є статистична оцінка ймо-

вірності загрози та оцінка відносної величини можливих втрат у успішній події загрози. Цей метод є розумним лише тоді, коли аналізована комп'ютерна система взагалі не захищена. Наприклад, коли є лише її проект. Якщо система існує певний час і використовує програмне забезпечення, яке містить певні механізми захисту, тоді слід розглянути залишкову оцінку ризику. У цьому випадку слід враховувати не тільки статистичну ймовірність реалізації загрози, але й ймовірність реалізації цієї загрози за наявності певних захисних заходів. По-друге, цей метод базується на припущенні, що статистична оцінка і-ї загрози, реалізована в КС з багаторівневою архітектурою, має високу надійність та високу надійність для будь-якої іншої системи, побудованої на подібній архітектурі. У цьому випадку це не так, оскільки особливість сіткових систем полягає в тому, що вони будуються з використанням деяких додаткових стеків протоколів на рівні програми моделі OSI, які мають власну багаторівневу архітектуру.

Існує ще один спосіб оцінки ризиків, пов'язаних із реалізацією інформаційних загроз, який є загальним на практиці та описаний у [13]. Цей метод використовує модель оцінки ризику для трьох факторів: загрози, вразливості та вартості збитків. Загроза стосується сукупності умов та факторів, які можуть призвести до порушення конфіденційності, цілісності або доступності інформації. Вразливість розуміється як слабкість системи захисту, яка може реалізувати певні загрози. Ймовірність подій за цим методом може бути об'єктивною або суб'єктивною. Об'єктивна ймовірність - це відносна частота події в загальній кількості спостережень або відношення кількості позитивних результатів до загальної кількості. Суб'єктивна ймовірність - це міра впевненості людини чи групи людей у тому, що подія відбудеться. В останньому випадку це експерт або представник експертної групи.

Експерти повинні мати належний досвід та знання щодо індивідуальних особливостей предметної області та побудови цієї конкретної системи. Імовірність реалізації загрози залежить від ймовірностей (у разі статистичного визначення) або рівнів загроз та вразливостей (у разі експертного судження),

які є незалежними значеннями. Рівень P_i реалізацій деякої загрози визначається за формулою [14]:

$$P_i = P_{\text{зі}} P_{\text{ві}},$$

де $P_{\text{зі}}$ – рівень (ймовірність) реалізації і-ї загрози, $P_{\text{ві}}$ – ступінь вразливості або легкості (ймовірність), з якою реалізація і-ї загрози може привести до негативних наслідків. Тоді залишковий ризик R_o визначається за формулою:

$$R_o = \sum_{i=1}^{N} P_{\text{зі}} P_{\text{ві}} Q_i,$$

Цей вираз можна вважати математичною формулою, якщо використовуються кількісні шкали, або як формулювання загальної ідеї на основі експертного судження, якщо хоча б одна із шкал є якісною. Дійсно, якщо змінні у цій формулі кількісні, тоді ризик - це оцінка математичного прогнозування збитків, якщо набір небезпек успішно реалізований. Якщо змінні категоричні, операція множення метрики не визначена, і ви не повинні використовувати цю формулу явно.

1.3. Оцінка кібербезпеки SMART GRID-систем

У світовому енергетичному секторі існують різні тлумачення поняття «розумні мережі» (Smart Grid). Загалом, «розумна» мережа - це електрична мережа, яка на основі сучасних інноваційних технологій ефективно координує та керує роботою всіх підключених об’єктів - від різних систем виробництва, передачі та розподілу електроенергії до одержувачів з метою створення економічно вигідної і стабільна енергія системи з низькими втратами та високим рівнем надійності та якості енергопостачання [4].

Згідно з Європейською технологічною платформою, Smart Grids - це «модернізовані мережі, які використовують інформаційно-комунікаційні мережі та технології для збору інформації, яка автоматично підвищує ефективність, надійність, стійкість виробництва та розподілу». На рис. 1.7 представлена загальна сучасна конфігурація системи Smart Grid [2].

З точки зору споживача, «розумні мережі» дають можливість активно контролювати споживання енергії, використовувати гнучкі енергетичні плани та навіть стати невеликими постачальниками електроенергії. Це дозволяє постачальникам енергії встановлювати ціни з урахуванням часу, покращувати планування потужності та споживання енергії, а також гнучкіше адаптуватися до потреб ринку. Мережа покращує управління передачею енергії та підвищує стійкість до відмов системи управління.

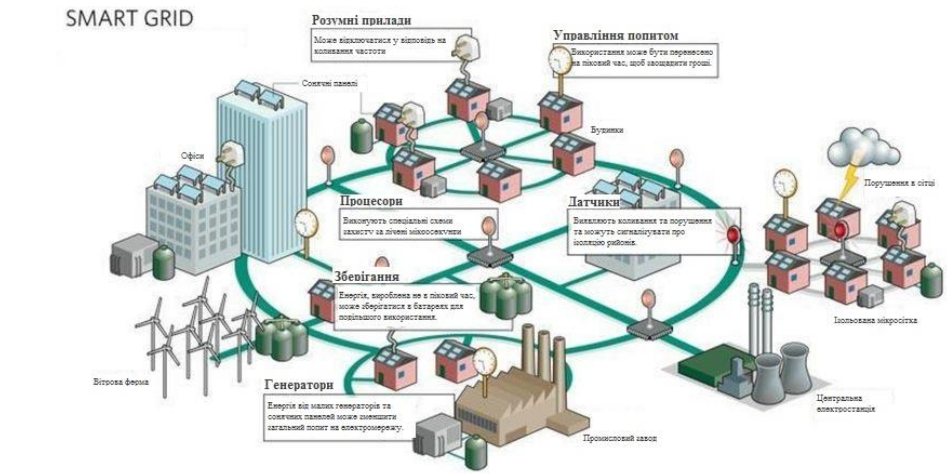

Рис. 1.7. Сучасна конфігурація Smart Grid системи.

Водночас інтенсивне використання інформаційно-комунікаційних технологій викликає багато нових проблем. «Розумні мережі» - це набір різноманітних застарілих систем, оточених новими технологіями та архітектурними підходами, що відповідають різним стандартам та нормам, які повинні поєднуватися в єдину мережу зв'язку. Інтелектуальні системи мережевого зв'язку мають багато слабких сторін, які відрізняються між мережами. Підключення «розумної мережі» до Інтернету піддає мережу новим типам загроз, включаючи розширені стійкі загрози (APT), розподілену заборонену службу (DDoS), ботнети та нульові дні, Stuxnet, Duqu, Red October та Black Energy лише декілька прикладів сучасних загроз, що виникли з 2010 року [4].

Впровадження «розумних мереж» вимагає мультидисциплінарного підходу, який поєднує різні технології та включає управлінські, політичні, юри-

дичні та інші аспекти. Важливою частиною цього процесу є оцінка безпеки, тобто оцінка рівня безпеки та виявлення потенційних вразливих місць, які можуть бути використані зловмисниками. У літературі можна знайти кілька визначень оцінки безпеки. У цій главі представлені визначення аналізованих стандартів. Визначення цих тез користуються найбільшим визнанням серед експертів з питань інформаційної безпеки. МЕК TS 62351-1 визначає оцінку безпеки як «обхідний процес оцінки активів з урахуванням вимог безпеки, заснований на ймовірному ризику нападу, відповідальності за успішні атаки та витратах на покращення ризику та підзвітності [5].

NIST SP 800-53 прирівнює оцінку безпеки до оцінки контролю безпеки та визначає її як «Випробування та/або оцінка засобів управління, оперативного та технічного контролю безпеки в інформаційній системі з метою визначення ступеня належного управління ними, функціонування за призначенням та даючи бажаний результат для забезпечення вимог безпеки системи. для системи» [6]. За даними Міністерства національної безпеки США (DHS), оцінки безпеки базуються на аналізі безпеки засобів управління в системі, «визначають засоби управління, які виконуються правильно, функціонують за призначенням і досягають бажаного результату для задоволення вимог безпеки системи» [7].

NIST SP 800-115 визначає оцінку інформаційної безпеки як "процес визначення ефективності конкретної цілі безпеки, яка оцінюється (наприклад, хост, система, мережа, процедура, особа, відома як оцінювач)". Стандарт виділяє три типи методів оцінки, які можуть бути використані для їх виконання [8]:

– тестування - аналіз об'єктів оцінки в конкретних умовах з метою порівняння фактичної та очікуваної поведінки;

– експертиза – перевірка, огляд, спостереження, вивчення або аналіз об'єктів оцінювання з метою з метою пояснення, розуміння або збору необхідних доказів;

– інтерв'ю – обговорення з окремими особами чи групами в організації для з’ясування, розуміння або встановлення місця доказування.

Визначення дуже схожі. Їх спільним знаменником є розуміння оцінки безпеки як процесу, заснованого на аналізі ресурсів для визначення того, чи відповідають вони вимогам безпеки (або цілям безпеки). NIST SP 800-115 розширює визначення, включаючи методи оцінки безпеки. Таким чином, оцінка безпеки - це процес визначення того, наскільки організація, що оцінюється, відповідає конкретним цілям безпеки або вимогам безпеки. Це можна зробити за допомогою трьох типів методів: тест, іспит та співбесіда.

Методи оцінки безпеки можна розділити на такі групи:

- Зворотній зв'язок - пасивне, як правило, ручне дослідження, що проводиться для виявлення вразливостей системи безпеки. Сюди входять перегляд документації, журналів, правил та конфігурацій, перевірка відповідності, офіційний аналіз, відслідковування мережі та перевірка цілісності файлів;
- Ідентифікація вразливості - ручне або автоматичне (зазвичай) виявлення системних несправностей. Методи ідентифікації включають виявлення мережі, сканування портів, сканування вразливостей, бездротове сканування та перевірку безпеки додатків;
- Аналіз вразливості - ручне або автоматичне дослідження виявлених вразливостей з метою остаточного підтвердження їх існування та розвитку подальших наслідків їх експлуатації. Методи включають злом паролів, злом, соціальну інженерію та тестування безпеки додатків;
- Тестування на відповідність визначає, чи відповідають системи цілям безпеки або вимогам безпеки. Мережеве нюхання - це інструмент для пасивного моніторингу мережевого спілкування та перевірки його вмісту, щоб перевірити, чи достатньо він захищений. Повторна перевірка цілісності файлу виявляє модифікації файлів на основі розрахунків контрольної суми;
- Формальний аналіз використовує формальну логіку, дискретну математику та інші математично обґрунтовані методи для оцінки без-

пеки інформаційних систем. Оцінка вимагає підготовки формальних специфікацій аналізованих систем, які згодом можуть бути перевірені, як і перевірка математичних формул. Формальні методи часто мають логічні обчислення, які можна систематично перевіряти за допомогою автоматизованого інструменту;

- Розкриття мережі - це розпізнавання структури мережі, яке зазвичай відбувається поза межами мережі. Сканування портів визначає відкриті комунікаційні порти, які часто є першою мішенню зловмисників. Сканування вразливості шукає (зазвичай відомі) вразливості програмного забезпечення. Це допоможе вам виявити застарілі версії програмного забезпечення, відсутні виправлення або неправильну конфігурацію. Бездротове сканування перевіряє, чи може зовнішній користувач отримати доступ до бездротових мереж або зв'язку;
- Злом паролів спрямований на виявлення паролів на основі наявних даних для виявлення слабких паролів та політики щодо паролів. Тестування на проникнення, «зв’язок червоної команди» або «злом білих капелюхів» або інші так звані процедури «етичного злому» використовують підходи до злому для аналізу вразливості системи. Соціальна інженерія полягає у тому, щоб примусити людей діяти, що піддало б систему хакерським атакам, перевірку процедур безпеки та поведінку користувачів системи (обізнаність користувачів);
- Експертиза у тестуванні та тестуванні безпеки підтверджує, що програмне забезпечення є вразливим, надійним та надійно працює з користувачами, іншими програмами та середовищем виконання. Процес оцінки безпеки «розумної мережі», як визначено в NIST SP 800-115, показаний на рис. 1.8, може бути використаний для оцінки кібербезпеки систем Smart Grid.

Аналіз показує, що критерії оцінки мережевої безпеки ще не визначені. Стандарти інформаційної безпеки "розумної мережі" вирішують цю проблему

по-різному і по-різному, але дають досить загальні рекомендації без технічних особливостей. Вони можуть використовуватися як керівництво для діяльності вищого рівня, наприклад, для визначення стратегій оцінки безпеки, розподілу відповідальності або планування дій з оцінки безпеки.

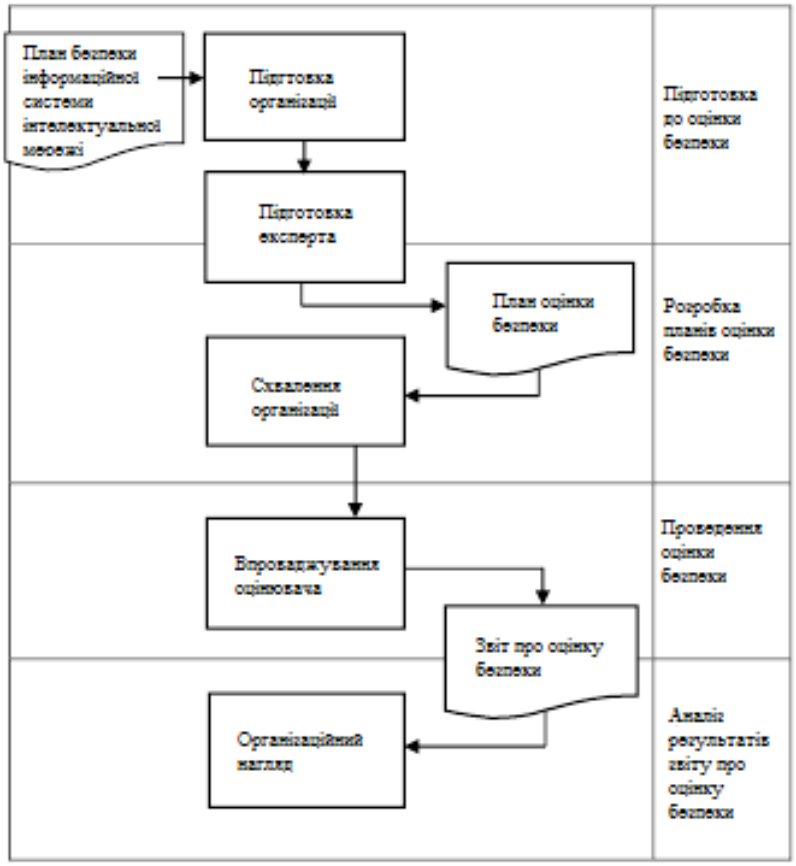

Рис. 1.8. Процес оцінки безпеки Smart Grid систем [8]

Дані стандарти можуть бути застосовані до корпоративного рівня «інтелектуальної мережі», а також до всіх її компонентів, що використовують комунікаційні технології та проводять обробку інформацію. Поміж них NIST SP 800-115 виділяється як найбільш повне джерело вказівок щодо оцінки безпеки.

РОЗДІЛ 2. СУЧАСНІ РІШЕННЯ В СФЕРІ МОНІТОРИНГУ

2.1. Характеристика сучасних методів моніторингу та виявлення атак

В даний час не існує універсального рішення проблеми комп'ютерних систем та інформаційних мереж, що виявляють ненормальні умови під час обробки інформації. Однак, з розвитком автоматизації інформаційних технологій та постійною модернізацією програмного та апаратного забезпечення комп'ютерної системи, вирішення проблеми виявлення аномалій вже не може гарантувати безпеку системи [8].

Метод виявлення аномалії (рис. 2.1) часто використовується для вирішення проблеми виявлення атак на комп'ютерні системи та інформаційні мережі. Вони вибираються на основі певного набору системних параметрів, і їх ефективність залежить лише від цього набору параметрів. З точки зору теорії розпізнавання образів, найбільш доцільно використовувати схеми прийняття рішень для класифікації.

Рис. 2.1. Класифікація методів виявлення аномалій

Метод підпису - це метод виявлення вторгнень, заснований на бібліотеці підписів, і найчастіше використовується в системі виявлення вторгнень, що містить підпис (шаблон) типової атаки, створений на основі заголовка або вмісту мережевого пакета. Якщо розглядати з точки зору обчислювальних витрат, велика кількість підписів зробить цей метод дорожчим. Для вирішення цього обмеження пропонується новий метод, що поєднує традиційне порівняння сигнатур та аналіз даних. Головною перевагою цього методу є підвищення продуктивності методу підпису та зменшення кількості помилкових спрацьовувань, оскільки пошук відбувається лише у певних частинах пакету.

Метод підпису має наступні переваги:

• ефективно виявляє атаки на інформаційні системи;

• відсутність великої кількості помилкових спрацьовувань;

• надійна оцінка використання конкретних інструментів або методів атаки;

• можливість точнішого встановлення параметрів підпису.

Недоліки методу підпису:

• базу даних підписів потрібно оновити для виявлення нових атак;

• неможливо виявити атаки, не описані в експертній системі;

• неможливість виявити атаки, що відрізняються від сигнатурного опису, або не мають опису.

Поведінкові методи - це методи, засновані на "нормальній" функціональній моделі інформаційної системи, а не методи, засновані на моделі інформаційної атаки. Принцип дії методу полягає у виявленні різниці між поточним режимом роботи інформаційної системи та режимом, що використовується як еталон для інформаційної системи. Будь-яка різниця вважається вторгненням або аномалією. Складність цього принципу полягає у створенні точної еталонної моделі.

Метод поведінки має наступні переваги:

• визначити атаку без конкретних деталей (підпис);

• детектори аномалій можуть генерувати інформацію, яка може бути використана для подальшої ідентифікації підписів атак;

• високо чутливий до змін стану інформаційної системи.

Недоліки поведінкових методів:

• помилкові сигнали, коли поведінка користувача непередбачувана;

• помилкові тривоги з непередбачуваною мережевою активністю;

• час, проведений на етапі навчання системи..

Сигнатурні методи:

Метод контекстного пошуку полягає у виявленні певного набору символів у вхідній інформації. Наприклад, щоб виявити атаку на веб-сервер під Unix-подібною операційною системою, маючи на меті отримати несанкціонований доступ до файлу паролів, шукайте певну послідовність символів «Get * / etc / password» в загловку запиту HTTP. Метод аналізу стану заснований на формуванні підписів атак, які мають форму низки переходів інформаційної системи з одного стану в інший. Збір таких подій встановлюється параметрами сигнатури атаки.

Поведінкові методи (методи виявлення аномалій):

Метод, заснований на статистичній моделі, визначається статистичними показниками, що характеризують поведінкові параметри правил системи. Якщо ці параметри з часом відхиляються від встановлених значень, реєструється виявлення атаки.

Комбіновані методи:

1. Метод виробничих правил дозволяє описати модель атаки природною мовою, з високим ступенем абстракції. Експертна система, що використовується у цьому методі, містить дві бази даних: факти та правила. Фактом є вхідні дані з інформаційної системи. Правило - це алгоритм, що використовується для прийняття логічних рішень щодо фактів нападу на основі сукупності фактів. Сформована система правил повинна описувати характеристики атак, які система повинна виявити.

2. База правил. Використовуючи експертну систему, можна точно визначити взаємодію між вузлами інформаційної системи, яка завжди здійснюється відповідно до певної домовленості. Якщо в процесі обміну інформацією з’являється невідома команда або нестандартне значення одного з параметрів між вузлами, це можна вважати ознакою атаки.

3. Метод моделювання поведінки біологічних систем використовує алгоритми та моделі на основі біологічних об'єктів, а саме генетичних алгоритмів та штучних нейронних мереж (ANN). Вважається, що моделі, засновані на біологічних моделях, є більш перспективними завдяки своїй пристосованості та саморозвитку.

На фазі вторгнення для виявлення атак можна використовувати підписи та поведінкові методи. Будь-яке вторгнення має певні характеристики, з одного боку, воно може мати форму підпису, з іншого боку - у формі відхилення від довідкової інформаційної системи поведінки. Найефективнішою є комбінація цих методів, і для того, щоб отримати необхідні вхідні дані, потрібно використовувати будь-який (мережевий або вузловий) датчик. Було зроблено багато спроб побудувати ADS. Більшість з них є концептуальними моделями, призначеними для перевірки того, чи можливо здійснити математичні моделі чи методи. Комерційних продуктів IDS небагато, а існуючі продукти майже ніколи не перевершували MDS.

Майже всі методи, що були описані можна розділити на:

а) ті, що базуються на зберіганні прикладів поведінки;

б) частотні;

в) нейромережеві;

г) ті, що будують скінченні автомати;

д) інші спеціальні.

Основне завдання - знайти найкращий спосіб побудови ADS в системі IDS [9]. Метод, заснований на прикладі збереженої поведінки, є найпростішим методом, тобто безпосереднім запам'ятовуванням прикладу операції, послідовності команд користувача або будь-яких параметрів, доступних при реєстрації.

Цей метод дуже ефективний у виявленні вторгнень, оскільки має обмежену кількість можливих дій основного корпусу комп’ютерної системи, важливу визначеність завдань, які можна виконати, і структуру операційної системи. Реакція комп’ютерної системи на ненормальну поведінку процесу змушує її сповільнюватися.

Отже, процес, який виявляє аномально активну поведінку, буде зупинений і автоматично знищений системою, оскільки він не відповів на запит. Попередньо записана послідовність системних викликів є частиною навчального набору і буде запам'ятовуватися під час роботи та перевірятись, чи існує вона в поточному сеансі. Через програми, що викликаються програмою моніторингу, через їх регулярність розмір бази даних підпослідовностей буде не дуже великим. Послідовність поточного заняття (а не між тренуваннями) вважається ненормальною. Цей метод вимагає вдосконалення ядра операційної системи, що не завжди можливо. Крім того, тривала присутність таких компонентів на моніторі може сповільнити роботу всієї системи на 4-50%.

Метод на базі частотної моделі

Розвиток системного мислення на основі конкретних випадків враховує частотний розподіл системних параметрів. Рекомендується зберігати інформацію про суб'єкта в режимі активності, тобто використовувати статистичні терміни для представлення поведінкових особливостей суб'єкта для конкретного об'єкта, таких як авторизація, запуск програми, доступ до файлів та реєстрація ненормальних пристроїв. Далі перевірте, чи не попадає відносна кількість певних подій за заданий інтервал. Модифікація частотного методу – це робота із запропонування методу, заснованого на структурних нулях. Цей метод передбачає використання інформації про команди, які рідко використовуються або взагалі не використовуються. У таблиці ймовірностей клітинки, що відповідають цим командам, дорівнюють нулю, тобто структура дорівнює нулю. Введіть унікальний індекс, який розраховується для кожного користувача та кожного сеансу. Для команд, які часто використовуються в поточному сеансі, індекс збільшиться позитивно. Чим менший індекс, тим вища частота використання цієї

команди. Тому широке використання команд потоку може призвести до високих значень індексу унікальності. У випадку некористувацьких команд індекс зменшується. Припускаючи, що значення індексу певної теми є стабільним, автор намагається розрізнити їх за значенням індексу.

Досить типовим недоліком частотного методу є його несумісність, оскільки еталонне значення частоти зазвичай визначається лише один раз, визначається навчальним набором або експертними даними, а порядок інструкцій ігнорується.

Метод на базі нейромережевої моделі

Використання нейронних мереж спричинене самою задачею виявлення ненормальних подій. Ідея полягає в наступному: отримавши деякі «навчальну» набори параметрів введення-виведення, що характеризують поведінку системи, що дозволяє мережі адаптуватися до них і розглядати їх як специфікації. Якщо вхідні дані є регулярними, передбачається, що мережа може вчитися на запропонованих шаблонах. Якщо в цьому процесі вихід є фактором, який потрапляє в небезпечну зону або відрізняється від реальної системи, припускаючи, що це один із параметрів системи, робиться висновок, що система ненормальна.

Для побудови шаблону поведінки користувача використовуються такі параметри: час роботи, набір вузлів, з якого починається робочий сеанс, та характеристики використання системних ресурсів. Ці параметри обробляються та вводяться в нейронну мережу зворотного поширення помилок. Початковий коефіцієнт для користувачів із нормальною поведінкою дорівнює нулю, а для користувачів із ненормальною поведінкою - один. Іншими словами, мережа навчається на парах типів («нормальний» параметр, 0) та («ненормальний» параметр, 1). Оскільки для того, щоб отримати «ненормальну» поведінку, користувача потрібно змусити виконувати ненормальну поведінку, випадкові дані генеруються випадковим чином, що ускладнює інтерпретацію результатів, пов'язаних з фактичними даними. Результатом є ідентифікатор користувача (параметр, хоча це число в UNIX, він буде штучно наближатись до користувача, ідентифікатор якого близький). Якщо користувача помилково ідентифіковано,

адміністратор отримає лист від системи. Якщо отриманий результат містить лише 10 користувачів, то ці умови є «теплицями», оскільки в реальній системі кількість користувачів може сягати тисяч, і більшість з них виконують однотипні операції, що змушує їх розрізняти таким чином. складні. Результати показують, що цей метод застосовний і ефективний, але нерегулярність поведінки користувачів значно збільшує ступінь помилок, таких як помилкові спрацьовування. Очікується, що кращі результати можуть бути отримані з більш традиційних джерел (таких як системні процеси).

Метод, що базується на моделях, які будують скінченні автомати

Порівняно із звичайною частотою та методами, заснованими на екземплярах, цей метод можна використовувати для отримання більшої аналогової ємності. Вхідні дані розглядаються як потік дискретних подій, таких як системні виклики або ідентифікатори процесів. Мета полягає в отриманні алгоритму для моделювання заданої послідовності подій. Для великої кількості послідовностей ймовірність наступного символу, елемента чи сигналу залежить від ймовірності попереднього символу, елемента чи сигналу є його характеристикою. Зазвичай вони покладаються лише на кілька попередніх. Це спонукає їх використовувати ланцюги Маркова для моделювання. Проте у разі зростання порядку ланцюгів, що може суттєво збільшити точність моделі, кількість станів відповідного автомата поводить себе як O(ΣL), де Σ – розмір алфавіту символів; L – порядок ланцюга. Це ставить великі вимоги до ресурсів і збільшує час обробки. За вхідними даними будується матриця переходів ланцюга першого порядку і ймовірність сесії визначається як добуток ймовірностей переходу між станами, що відповідають елементарним подіям у файлі аудиту.

Другий метод використовується для побудови імовірнісного автомата з набором входів-подій з потоку аудиту та набором виходів - послідовністю чисел, що вказують на ступінь аномалії вхідної події. Ці цифри показують кореляцію між ймовірністю існуючого переходу з поточного стану в наступний стан та вектором ймовірності переходу з цього стану. Хоча для цього ступеня відхи-

лення було зроблено кілька необґрунтованих виборів, експериментальний метод виявився ефективним.

Інші методи

На додаток до вищезазначених методів, байєсівські мережі також використовуються для виявлення аномалій: це графічні моделі, що представляють взаємозв'язок між об'єктами та розподіл ймовірностей у власній структурі. Кореляція між станами оцінюється на основі набору навчальних даних та мережі із підключеними та взаємозалежними вузлами, ймовірності переходу між станами, які потрібно заповнити, та мережі вузлів, підключених до таблиці розподілу ймовірностей цих станів вузлів. .предок. Модель не є адаптивною, оскільки її структура суворо залежить від наданих навчальних даних. Візьміть набір двійкових ліній фіксованої довжини як зразки. Кількість рядків у колекції також фіксована.

В процесі еволюції шляхом адаптації вибирається рядок символів, що створює рядок символів наступного покоління. Наступне покоління засноване на використанні трьох основних операторів рядків: виділення (виберіть найбільш підходящого представника), копіювання (обмін частинами рядка) та мутації (випадкова зміна бітів у рядку для запобігання безповоротній втраті) інформації). Генетичні алгоритми використовуються для пошуку песимістичних сценаріїв у змішаній проблемі виявлення зловживань та аномалій. З одного боку, виявляються лише відомі атаки, з іншого боку, використання генетичних алгоритмів та суворі обмеження припущень також можуть фіксувати зміни в цих атаках. Цей метод перевіряється на штучних даних шляхом моделювання простих атак.

Отже, у власне IDS, на основі аналізу вищезазначених методів, може бути рекомендована комбінація різних методів, і на основі цього можна зробити остаточний висновок щодо існування та характеру втручання [10]. Моделі цих

методів не повинні покладатися на конкретні типи аудиторських даних, і тоді вони можуть застосовуватися до будь-якої послідовності аудиту. Отже, такими даними, крім команд, безпосередньо виконуваних користувачем, можуть бути також специфічна для програми послідовність системних викликів, значення інтервалу між натисканнями клавіш на клавіатурі, інтервал між робочими сеансами та значення послідовності курсору миші Позиція, сортування значень полів заголовка мережі всіх рівнів або значень, які не включені, але залежать від них. Отже, будь-який сигнал, команда, елемент або послідовність часу чи інші значення, характерні для користувача, процесу, системи або сегмента мережі, можуть бути використані в ADS.

Насправді користувачі схильні змінювати свою поведінку (змінюючи завдання, набуваючи нових навичок), тому метод моделі повинен бути адаптованим. Досить багато атак спрямовано на системні програми, такі як отримання дозволів суперкористувача та використання вразливостей у програмі SUID. Зазвичай після цього програма виявляє зовсім інший характер від системного дзвінка, який спостерігався до атаки. Тому варто звернути увагу на виявлення аномалій рівня системного виклику, яке стає все більш популярним, оскільки дозволяє абстрагуватися від нерегулярної поведінки людини. У той же час ми не можемо відмовитись від аналізу даних аудиту користувачів, оскільки лише на цьому рівні можна виявити вторгнення, які не відбулися на рівні системного дзвінка (наприклад, використання викрадених паролів).

Повноцінна IDS повинні включати компоненти виявлення зловживань, оскільки атаки вразливості все ще є основним типом атак сьогодні. Але є багато причин, чому метод нейронної мережі є найкращим, хоча його реалізація вимагає значних зусиль та адаптації до потреб бізнесу.

2.2. Класифікація засобів моніторингу та аналізу

Постійний моніторинг роботи локальної мережі є основою мережевої безпеки будь-якої компанії і необхідний для належної роботи. Контроль - це набір необхідних заходів, які необхідно виконувати під час управління мережею. Через важливість цієї функції вона часто відокремлена від інших систем управління та реалізується спеціалізованими установами. Таке розділення функцій управління та самоуправління дуже корисно для малих та середніх мереж, де економічно недоцільно встановити інтегровану систему управління.

Використання незалежного контролю допомагає адміністраторам мережі самостійно виявляти проблемні області та досліджувати топологію мережі, маючи при цьому можливість вручну виконувати відключення або реконфігурацію. Процес управління мережею ділиться на два етапи - аналіз та моніторинг [11]. На етапі моніторингу збираються основні дані роботи мережі: підраховується кількість обмінів кадрів і пакетів різних протоколів у мережі, а також стан порту концентраторів, комутаторів та маршрутизаторів. Далі виконується аналіз - це процес розуміння інформації, зібраної на етапі моніторингу, порівняння її з даними, отриманими раніше, і припущення можливих причин повільної або ненадійної роботи мережі.

Таблиця 2.1. Основні засоби моніторингу

Засіб моніторингу	Характеристика
Системи керування мережею (NetworkManagementSystems)	централізовані програмні системи, які збирають дані про стані вузлів і комунікаційних пристроїв мережі, а також дані про трафік, який циркулює в мережі. Ці системи не тільки здійснюють моніторинг і аналіз мережі, а й виконуються в автоматичному чи полуавтоматичному режимі дії щодо управління мережею - включення і відключення портів пристроїв, зміна параметрів мостів адресних таблиць мостів, комутаторів і маршрутиза-

	торів тощо. Прикладами систем управління можуть служити популярні системи HPOpenView, SunNetManager, IBMNetView.

Продовження табл. 2.1

Засоби управління системою (SystemManagement)	Засоби управління системою часто виконують функції, аналогічні функціям систем управління, але стосовно інших об'єктів. У першому випадку об'єктом управління є програмне і апаратне забезпечення комп'ютерів мережі, а у другому - комунікаційне устаткування.
Вмонтовані системи діагностики і управління (Embeddedsystems)	Ці системи виконуються у вигляді програмно-апаратних модулів, встановлюваних у комунікаційне устаткування, а також у вигляді програмних модулів, вмонтованих в операційні системи. Вони виконують функції діагностики і управління лише одним устроєм, і в цьому їх основна відмінність від централізованих систем управління.

Аналізатори протоколів (Protocolanalyzers).	Представляють собою програмні чи апаратно-програмні системи, які обмежуються на відміну від систем управління лише функціями моніторингу аналізу трафіку мережах. Хороший аналізатор протоколів може захоплювати і декодувати пакети великої кількості протоколів. Аналізатори протоколів дозволяють встановити деякі логічні умови для захоплення окремих пакетів і виконують повне декодування захоплених пакетів.

Продовження табл. 2.1

Устаткування для діагностики і сертифікації кабельних систем.	Умовно це устаткування можна поділити на чотири основні групи: мережні монітори, прилади для сертифікації кабельних систем, кабельні сканери і тестери (мультиметри).
Експертні системи.	Цей вид систем акумулює людські знання про виявленні причин аномальної роботи мереж і можливих засобів приведення мережі в працездатний стан. Експертні системи часто в вигляді окремих підсистем різних засобів моніторингу аналізу мереж: систем управління мережами, аналізаторів протоколів, мережевих аналізаторів.
Багатофункціональні пристрої	Останніми роками, у зв'язку з повсюдним

аналізу та діагностики.	поширенням локальних мереж виникла необхідність розробки недорогих портативних приладів, які сполучають функції кількох пристроїв: аналізаторів протоколів, кабельних сканерів і, навіть, деяких можливостей ПЗ мережного управління.

За допомогою програмних та апаратних тестерів, мережевих аналізаторів, вбудованих комунікаційних пристроїв та агентів системи управління вирішуються завдання моніторингу. Завдання аналізу вимагають активної участі людей та використання складних інструментів, таких як експертні системи.Багато спеціалістів мережі вже набули практичного досвіду. Всі методи моніторингу та аналізу мережі можна розділити на кілька категорій:

Системи управління мережею (Network Management Systems) – це централізована програмна система, яка використовується для збору даних про стан мережевих вузлів та обладнання зв'язку та даних про мережевий трафік. Ці системи не тільки виконують моніторинг та аналіз, але також виконують автоматичні або напівавтоматичні режими ввімкнення та вимкнення портів пристроїв управління мережею, зміну параметрів таблиці адрес мостів, комутаторів, маршрутизації перевантажень тощо. Прикладами систем управління є популярні системи HPOpenView, SunNetManager, IBMNetView.

Засоби управління системою (System Management). Об'єктами управління є програмне та апаратне забезпечення мережевих комп'ютерів, а також комунікаційне обладнання. Крім того, деякі функції цих двох систем управління можуть дублюватися.Наприклад, засоби управління системою можуть виконувати найпростіший аналіз мережевого трафіку. Найвідоміші системи управління системою включають LANDesk, IBM Tivoli, Microsoft Systems Management Server, HP OpenView, Novell ZENworks та CA Unicenter.

Вбудовані системи діагностики і управління (Embedded Systems). Така система реалізована у вигляді програмно-апаратних модулів, встановлених в

комунікаційному пристрої, і у вигляді програмних модулів, вбудованих в операційну систему. Вони використовують лише один пристрій для виконання функцій діагностики та контролю, що є основною відмінністю між ними та централізованими системами управління. Прикладом такого інструменту є модуль управління концентратором Distrebuted 5000, який реалізує автоматичне розподіл портів при виявленні несправностей, розподіл портів на внутрішній сегмент концентратора та інші ситуації. Зазвичай вбудований модуль управління також діє як агент SNMP для передачі даних про стан пристрою в систему управління.

Аналізатори протоколів (Protocolanalyzers). Вони являються програмними або апаратними системами, на відміну від систем управління, вони лише здійснюють контроль та аналіз мережевого трафіку. Найкращий аналізатор протоколів може захоплювати і декодувати пакети даних великої кількості протоколів, що використовуються в мережі, - як правило, їх десятки. Ці інструменти дозволяють встановити певні логічні умови для захоплення одного пакета даних і виконати повне декодування захопленого пакета даних, тобто відобразити дешифрування самого пакета даних, вбудованого в різні рівні протоколів, і вміст кожного поля у зручній формі.

Обладнання, призначене для проведення діагностики та сертифікації кабельної системи. Існує чотири групи: мережевий монітор, пристрій сертифікації кабелю, сканер кабелю та тестер (мультиметр). Мережеві монітори використовуються для тестування різних типів кабелів. Варто розрізняти монітори мережі та аналізатори протоколів. Мережеві монітори збирають дані лише про статистику трафіку – середній загальний мережевий трафік, пакетний трафік з певними типами помилок тощо. Сертифікація проводиться відповідно до вимог одного з міжнародних стандартів. Кабельні сканери використовуються для діагностики мідних кабельних систем. Тестер призначений для перевірки кабелю на наявність фізичних несправностей.

Експертні системи. Ці системи являють собою набір методів, що використовується для виявлення причини ненормальної роботи мережі та можливих

методів відновлення нормальної її роботи. Експертні системи зазвичай реалізуються у формі незалежних підсистем з різними методами моніторингу та аналізу мережі: система управління мережею, аналізатор протоколів та аналізатор мережі. Найпростіша версія експертної системи - це контекстно-залежна система довідки. Більш складна система - це база знань, що містить елементи штучного інтелекту. Прикладом такої системи є експертна система, вбудована в систему управління спектром Cabletron.

Багатофункціональне обладнання для аналізу та діагностики. Через поширення локальних мереж виникає потреба у розробці недорогих портативних пристроїв, що поєднують в собі безліч функцій пристроїв: аналізатори протоколів, кабельні сканери та навіть певні функції програмного забезпечення для управління мережею. Прикладом цього типу обладнання є Compas від Microtest Inc. або 675 LANMeter від FlukeCorp.

У процесі проектування мережі часто необхідно кількісно визначити деякі характеристики мережі, такі як інтенсивність потоку даних по лінії зв'язку, затримка на різних етапах обробки пакетів даних, час відгуку на один або інший запит і частота. Події та інші особливості. Для цих цілей можуть бути використані різні інструменти, найважливіший з яких - це засоби моніторингу в системах управління мережею, про які йшлося в попередньому розділі. Певні вимірювання мережі також можуть виконуватися програмами, вбудованими в операційну систему. Але найдосконалішим методом дослідження мережі є аналізатор протоколів [12].

Аналізатор мережевого протоколу можна використовувати для:

• орієнтування на складні питання;

• виявлення та ідентифікування несанкціонованого програмного забезпечення;

• отримання такої інформації, як базові схеми руху та показники використання мережі;

• визначення невикористаних протоколів, щоб видалити їх із мережі;

• формувння трафіку для тестування на проникнення для перевірки системи захисту;

• співпраці з системою виявлення вторгнень (IDS);

• прослуховування трафіку, тобто використання миттєвих повідомлень (IM) або бездротової точку доступу-точка доступу (AP), щоб знайти несанкціонований трафік;

• проведення мережевих досліджень.

Аналізатор протоколів - це незалежний виділений пристрій або персональний комп'ютер, як правило, портативний, оснащений спеціальною мережевою картою та відповідним програмним забезпеченням. Мережева карта та програмне забезпечення, яке використовується, повинні відповідати топології мережі (кільце, шина, зірка). Аналізатор підключений до мережі так само, як і звичайний вузол. Відмінність полягає в тому, що аналізатор може приймати всі пакети даних, що передаються по мережі, тоді як звичайна станція адресована лише йому.

Для підтримки роботи мережевого адаптеру та декодування отриманих даних, програмне забезпечення аналізатора складається з основного та додаткового програмного коду. Код програми залежить від типу топології тестованої мережі. Крім того, багато програм декодування зосереджені на конкретних протоколах, таких як IPX. Деякі аналізатори можуть також включати експертну систему, яка може порадити користувачеві, які експерименти проводити в тій чи іншій ситуації, що можуть означати результати вимірювань та способи усунення певних типів відмов мережі. Незважаючи на відносну різноманітність аналізаторів протоколів на ринку, ми можемо перерахувати кілька особливостей, які до певної міри є спільними для всіх них:

Інтерфейс користувача

Значна кількість аналізаторів має повноцінний та зручний інтерфейс, як правило заснований на Motif або Windows. Такий інтерфейс дозволяє користувачам:

• відображати результати аналізу інтенсивності трафіку;

• отримувати миттєві та середні статистичні оцінки продуктивності мережі;

• вимагати певних подій та критичних ситуацій для відстеження їх виникнення;

• декодувати різні рівні протоколів та подавати дані у чіткій формі.

Буфер захоплення

Різні аналізатори мають різні обсяги буфера. Буфер може бути розміщений на мережевій карті, або може бути виділено простір в оперативній пам'яті одного з комп'ютерів у мережі. Якщо буфер розташований на мережевій карті, він управляється апаратно, тому швидкість введення збільшиться. Однак це може зробити аналізатор дорожчим. Якщо продуктивність програми захоплення буде недостатньою, деяка інформація буде втрачена, а аналіз буде неможливим. Розмір буфера визначає можливість аналізу більш-менш репрезентативної вибірки захоплених даних. Але яким би великим не був буфер захоплення, він рано чи пізно заповниться. У цьому випадку захоплення зупиняється або заповнюється з початку буфера. Він може вимірювати середній трафік у сегменті локальної мережі, де встановлений мережевий адаптер аналізатора. Виміряйте коефіцієнт використання сегмента, матрицю вузлів перехресного потоку та кількість хороших і поганих кадрів, що проходять через сегмент. Можливість роботи з кількома агентами, які передають захоплені пакети даних з різних частин локальної мережі. Ці агенти найчастіше взаємодіють з аналізатором протоколів відповідно до власного протоколу прикладного рівня.

Фільтри

Фільтри дозволяють контролювати процес збору даних, тим самим економить буферний простір. Відповідно до значення певних полів пакетів, зазначених як умови фільтрації, пакет буде проігноровано або записано в буфер захоплення. Використання фільтрів значно пришвидшує та спрощує аналіз, оскільки позбавляє потреби переглядати непотрібні на даний момент пакети. Комутатори - це деякі умови, що надаються користувачем для запуску та зупинки процесу збору даних із мережі. Такими умовами можуть бути вико-

нання ручних команд для запуску та зупинки процесу захоплення, тривалість процесу захоплення, поява певних значень у кадрі даних. Комутатори можна використовувати разом із фільтрами, дозволяючи більш детальний і тонкий аналіз та більш ефективне використання обмеженої кількості буферів захоплення.

Пошук

Серед аналізаторів, виділяють ті, які озволяють автоматизувати перегляд інформації, що міститься в буфері, та пошук даних у ньому за певними критеріями. Поки фільтри перевіряють вхідний потік за умовами фільтрації, функції пошуку застосовуються до даних, які вже зберігаються в буфері.

Багатоканальність

Також, серед аналізаторів є такі, які дозволяють одночасно записувати пакети від безлічі мережевих адаптерів, що зручно для порівняння процесів, що відбуваються в різних сегментах мережі. Можливість аналізу мережевих проблем на фізичному рівні аналізаторів протоколів є мінімальною, оскільки вони отримують всю інформацію від стандартних мережевих адаптерів. Тому вони надсилають та узагальнюють інформацію про фізичний рівень, передану їм мережевою картою, і це значною мірою залежить від типу мережевої карти. Деякі мережеві адаптери повідомляють більш детальну інформацію про помилки кадрування та інтенсивність зіткнень на сегменті, а деякі взагалі не повідомляють таку інформацію вищим рівням протоколів, на яких запущений Аналізатор протоколів.

Метод аналізу може бути представлений у вигляді наступних етапів [13]:

• Аналізатор протоколів контролює мережевий трафік;

• Аналізатор працює на хост-станції.

Коли аналізатор запускається в хаотичному режимі (безладний режим), драйвер мережевого адаптера перехоплює весь трафік, що проходить через нього. Аналізатор протоколів передає перехоплений трафік на механізм декодера пакетів, який ідентифікує і «розщеплює» пакет на відповідному рівні ієрархії. Програмне забезпечення аналізатора протоколів перевіряє пакети даних та

відображає інформацію про них на головному екрані вікна аналізатора. Відповідно до функції конкретного продукту, подану інформацію можна додатково проаналізувати та відфільтрувати. Як правило, вікно аналізатора протоколів складається з трьох областей.

У верхній області відображаються зведені дані перехопленого пакету даних. Як правило, ця область відображає найменшу кількість полів, а саме:

• дата та час (у мілісекундах) перехоплення пакету даних;

• IP-адреси джерела та призначення;

• адреса джерела та адреса призначення порту;

• тип протоколу (мережевий, транспортний або прикладний рівень);

• деякі зведені відомості про перехоплені дані.

Середня область відображає детальну інформацію про пакет відповідно до моделі мережі OSI. Нарешті, в нижній області пакет даних представлений у шістнадцятковій або символічній формі-ASCII.

Апаратні засоби

Обладнання, що застосовується для діагностики та сертифікації кабельних систем. Кабельна мережа (дротова мережа, лінія зв'язку) – це мережа, що має в якості елементів кабельні лінії та компоненти. Кабельний вузол включає в себе все пасивне комутаційне обладнання, що використовується для кабельного підключення або фізичного завершення. Як правило, обладнання, що використовується для діагностики кабельної системи, можна розділити на три категорії: мережеві аналізатори, кабельні сканери та тестери [14]. Щоб правильно вибрати обладнання, потрібно визначитися з транспортним засобом. Перш ніж проводити більш детальний огляд цих пристроїв, ми надаємо необхідну інформацію про основні електромагнітні властивості кабельної системи [15].

Основні електромагнітні характеристики кабельних систем

Основними електричними характеристиками, що впливають на роботу кабелю, є: загасання, імпеданс (хвильовий опір), поперечні провідники двох пар скручених пар і рівень зовнішнього електромагнітного випромінювання. Перехресне посилання між скрученою парою або наближеними перехресними

зв’язками (NEXT) - це результат перешкод електромагнітного сигналу, що виникають у двох скручених парах. Одна кручена пара передає сигнал, а інша приймає сигнал. При передачі сигналу по кабелю, наприклад при передачі, в кабелі є перехресне посилання, яке приймає сигнал. Значення NEXT залежить від частоти передавального сигналу - чим вище значення NEXT, тим краще (для категорії 5 NEXT має бути не менше 27 дБ на 100 МГц, а для кабелів категорії 3 на 10 МГц, NEXT має бути на не менше 26 дБ). Затухання - це втрата амплітуди електричного сигналу при його поширенні по кабелю. Є два основних джерела загасання: електричні властивості кабелю та поверхневі ефекти. Останнє пояснює залежність загасання від частоти. Загасання вимірюється в децибелах на метр. Для кабелів категорії 5 на 100 МГц загасання на 100 м не повинно перевищувати 23,6 дБ. Для кабелів категорії 3 згідно з IEEE 802.3 10BASE-T допустиме значення загасання на сегменті 100 м не повинно перевищувати 11,5 дБ. дорівнює 10 МГц. Опір (імпеданс) - це загальний (активний і реактивний) опір в ланцюзі. Опір вимірюється в Омах, що є відносно постійним значенням для кабельних систем. Для неекранованих витих парних кабелів найпоширенішими значеннями опору є 100 і 120 Ом. Типове значення імпедансу Ethernet по коаксіальному кабелю становить 50 Ом, тоді як типове значення імпедансу мережі Arcnet становить 93 Ом.

Раптові зміни імпедансу по довжині кабелю можуть спричинити процеси внутрішнього відбиття, що може призвести до стоячих хвиль. Стояча хвиля - це коливання в суцільному середовищі, при якому кожна точка середовища періодично рухається з постійною амплітудою відповідно до свого положення. Стоячі хвилі не передають енергію. Робоча станція, підключена до кабелю поблизу вузла стоячої хвилі, не зможе приймати надіслані на неї повідомлення. Активний опір - це опір постійному струму в ланцюзі. На відміну від імпедансу, активний опір не залежить від частоти, але збільшується зі збільшенням довжини кабелю. Для неекранованих кабелів витої пари категорії 5 активний опір на 100 м не повинен перевищувати 9,4 Ом.

Ємність - це характеристика металевого провідника для накопичення енергії. Два електричних провідника в кабелі, розділених діелектриком, є конденсатором, який може накопичувати заряд. Ємність є непопулярною цінністю, тому вона повинна бути якомога меншою. Висока ємність кабелю спотворить сигнал і обмежить пропускну здатність лінії. Для кабельних систем категорії 5 значення ємності на 100 м не повинно перевищувати 5,6 нФ. Рівень зовнішнього електромагнітного випромінювання або електричного шуму є небажаною зміною напруги в провіднику. Існує два типи електричного шуму: фоновий і імпульсний. Електричні шуми також можна розділити на низькочастотні, проміжні та високочастотні. Джерела фонового електричного шуму варіюються від ліній електропередач, телефонів і флуоресцентних ламп до 150 кГц; комп'ютери, принтери, копіювальні апарати в діапазоні від 150 кГц до 20 МГц; і в діапазоні від 20 МГц до 1 ГГц - телевізійні та радіопередавачі, мікрохвильові печі.

Основними джерелами імпульсних електричних шумів є двигуни, вимикачі та зварювальні пристрої. Електричний шум вимірюється в мВ. Кабельні системи з витою парою не сильно піддаються дії електричних шумів (на відміну від NEXT). Мережевий аналізатор Мережевий аналізатор (не плутати з Аналізатором протоколів) - це еталонний засіб вимірювання, який використовується для діагностики та сертифікації кабельних та електропроводних систем. Прикладом є мережеві аналізатори Hewlett Packard - HP 4195A та HP 8510C.

Мережевий аналізатор включає високоточний генератор частоти та вузькосмуговий приймач. Передаючи сигнали різних частот в передавальній парі та вимірюючи сигнали в приймальній парі, можна виміряти загасання та NEXT. Мережеві аналізатори - це велике та дороге обладнання, призначене для використання в лабораторіях спеціально навченими техніками.

Кабельні сканери

Ці пристрої дозволяють визначити довжину кабелю, NEXT, ослаблення, імпеданс, електричні схеми, рівні електричного шуму та оцінити результати. У цій категорії є багато пристроїв, таких як сканери від Microtest Inc., Fluke Corp.,

Datacom Technologies Inc., Scope Communication Inc .. На відміну від мережевих аналізаторів, сканерами можуть користуватися не тільки спеціально навчені техніки, але навіть адміністратори-початківці. Метод "кабельного радіолокатора" або рефлектометр часової області (TDR) використовується для визначення місця несправності кабельної системи (обрив, коротке замикання, неправильно встановлений роз'єм тощо). Суть цього методу полягає в тому, що сканер посилає короткий електричний імпульс на кабель і вимірює час затримки до надходження відбитого сигналу.

Полярність відбитого імпульсу визначає характер пошкодження кабелю (коротке замикання або обрив). У правильно встановлених і підключених кабелях відбиті імпульси повністю відсутні. Точність вимірювання відстані залежить від знання швидкості поширення електромагнітної хвилі в кабелі. У різних кабелях воно буде різним. Швидкість поширення електромагнітних хвиль у кабелях (NVP) зазвичай встановлюється як відсоток від швидкості світла у вакуумі. Сучасні сканери включають електронні таблиці NVP для всіх основних типів кабелів і дозволяють користувачам самостійно встановлювати ці параметри після попереднього калібрування. Найвідомішими виробниками компактних кабельних сканерів є MicrotestInc., WaveTekCorp., ScopeCommunicationInc.

Тестери

Тестер кабельної системи - це найпростіший і найдешевший пристрій для діагностики кабельних з'єднувачів, який дозволяє визначити пошкодження кабелю, але на відміну від сканера кабелю, він не може визначити місце несправності.

Програмно-апаратні рішення

Основними ініціаторами переходу до відкритого Інтернету є такі компанії, як Google, Amazon, Facebook та Microsoft. Вони вже давно усвідомили всі переваги, особливо зменшення витрат, спричинене їх власним сервером розробки та виробництвом тайваньської компанії (цей вид товару називається «білою скринькою (white box)»). Реалізація подібних моделей мережевого

обладнання - це лише питання часу [16]. Традиційно основні вузли мережевої інфраструктури, комутатори та маршрутизатори з'являються перед клієнтами у таких формах: власне обладнання, власні мережеві операційні системи, вбудовані набори функцій та спеціальне програмне забезпечення.

Рис.2.2. Запропоновані рішення типу "white box"

Інтернет-гіганти розробляють власне мережеве обладнання (зліва – модульний комутатор Facebook, рис. 2.2) для просування відкритої платформи традиційних виробників рішень (картинка зправа - комутатори серії Dell Networking ON - від Open Networks, рис. 2.2). На відміну від вертикально інтегрованих власних мейнфреймів, сучасні обчислювальні рішення розбиваються протягом десятиліть. Клієнти можуть придбати апаратні платформи та сервери відомих брендів (Dell, HP, Lenovo) та постачальників білих скриньок (Quanta Computer, Supermicro). У цьому випадку операційну систему та програми можна придбати окремо від інших постачальників та встановити на апаратній платформі. Це розкладання стимулює інновації на всіх рівнях IT-стеку: процесори та інші апаратні компоненти, операційні системи та додатки. Однак, з точки зору архітектури обладнання, мережева індустрія перестала розвиватися на багато років десь на початку 1990-х [17].

Самі програмні та апаратні платформи мережевого обладнання, включаючи виділені чіпсети ASIC, традиційно постачаються вертикально інтегрованими постачальниками, що не знижує ціни та не заважає інноваціям. Але з проникненням у мережевий світ ідей програмованої мережевої інфраструктури (SDN) ситуація почала швидко змінюватися, і додатковими каталізаторами є такі продукти, як bare metal, white box і brite box.

2.3. Методи оцінки критичності вразливостей

Термінологія, що використовується для аналізу загроз, вразливостей та ризиків, не однакова. Багато символів мають кілька визначень або можуть використовуватися по-різному. Це пояснюється особливістю їх використання із використанням існуючих моделей та методів. Деякі з них не розрізняють загрози та ризики і використовують ці поняття як взаємозамінні, наприклад модель Swedish Rescue Services Agency [27].

Існує багато систем оцінки вразливості, створених комерційними та некомерційними організаціями. Деякі з них:

- CERT/CC;
- Система оцінки від Microsoft;
- Система аналізу вразливостей SANS;
- CVSS.

Кожна з цих систем має свої переваги, але вони мають одну різницю - ознаку вимірювання. Наприклад, CERT / CC використовує значення балів від 0 до 180 з урахуванням таких факторів:

- Чи схильна Інтернет-інфраструктурадо ризику;
- Який тип передумов необхідний для експлуатації вразливості.

У свою чергу, система аналізу вразливості SANS буде розглядати конфігурацію знайденої вразливості як стандартну чи ні, незалежно від того, є система клієнтом чи сервером. Рейтингова система Microsoft намагається відобразити складність експлуатації та загальний вплив використання цієї вразливості. Ці рейтингові системи є корисними, але вони представляють метод, при якому люди вважають, що наслідки використання вразливостей однакові для приватних осіб та компаній [28].

SANS при аналізі вразливостей (SANS Critical Vulnerability Analysis) присвоює критичний рівень в тому випадку, коли існує загальнодоступна програма, що використовує вразливість, або її використання не потребує спеціальних навиків. В іншому випадку навіть потенційно ризиковані проблеми матимуть високий, але не серйозний рівень ризику. Окрім простоти використання, оцінка вразливості також враховує поширеність вразливих систем, які використовують метод SANS [29].

Команда PSS корпорації Майкрософт використовує метод оцінки ризиків, пов’язаних із шкідливим програмним забезпеченням (тобто атаками, які використовують вразливості). Він розглядає існування оновлень, що усувають помилки, кількість векторів, які може використовувати зловмисник, і всюдисущість вразливих систем. Наприклад, «надзвичайно небезпечний хробак» повинен поширюватися через не оновлену вразливість програмного забезпечення Microsoft, використовуючи два або більше векторів атак у загальнодоступних системах.

CERT використовує методику розрахунку ступеня ризику вразливості, що залежить від наступних критеріїв [29]:

- наскільки доступною є інформація про вразливість?
- чи зареєстровані випадки використання вразливості?
- яка кількість вузлів мережі є вразливою?
- які можуть бути наслідки використання вразливості?
- наскільки легко використати вразливість?
- які умови використання вразливості?

На жаль, не існує офіційного визначення взаємозв'язку між умовами, їх можливим значенням та результуючим ступенем ризику, що залишає велику різницю в оцінках тієї самої вразливості.

Загальна система оцінки вразливостей (CVSS) – це відкрита схема, котра дозволяє здійснювати обмін інформацією про IT-вразливості. Система оцінки CVSS складається з таких показників: базові показники, часові показники та контекстні показники. Кожен із них - це число (бал) від 0 до 10 і вектор - корот-

кий опис, що містить значення, що використовується для отримання балу. Базові показники відображають основні характеристики вразливості. Тимчасовий - Відповідає характеристикам вразливості, які змінюються з часом. Контекст - характеристика, унікальна для середовища користувача.

2.4. Порівняльна характеритсика програмних засобів на ринку

Хоча інтерес до інформаційної безпеки зростає, процедур оцінки вразливості небагато. Однак завдання вибору найкращого товару є непростим завданням, оскільки аналіз повинен враховувати багато факторів.

Поміж іншого, існує велика кількість сканерів на вразливість. Сканер вразливості - це програмне чи апаратне забезпечення, що використовується для виконання діагностики та моніторингу мережевих пристроїв. Вони дозволяють сканувати мережу, комп’ютери та програми, щоб знайти потенційні проблеми безпеки, а також оцінити та усунути вразливі місця [30].

За допомогою сканерів, можливо виконати перевірку різних додатків, що знаходяться в системі, на наявність «дірок», якими можуть скористатися зловмисники.

Більшість сучасних сканерів безпеки дотримуються наступних принципів [30]:

- збір інформації про мережу, ідентифікація всіх активних пристроїв та сервісів, що запущені на них;
- виявлення потенційних вразливостей;
- підтвердження обраних вразливостей;
- автоматичне усунення вразливостей.

Таким чином, сканери застосовуються для того, аби вирішити наступні питання [30]:

- ідентифікація та аналіз вразливостей;

– інвентаризація ресурсів, таких як операційнв система, програмне забезпечення та пристрої мережі;

– формування змітів, які містять опис вразливості та можливі варіанти їх усунення.

"Аналіз вразливості" відноситься до механізму, який запускає модельовану атаку для проведення тесту на вразливість. Зондування використовує методи реалізації атак, для того, аби виявити чи підтвердити проблему [30].

Аналізуючи ринок програм, що допомагають проаналізувати ступінь вразливості, було виявлено, що дві утиліти можуть приблизно оцінити ступінь критичності.

Класифікація програмного забезпечення для оцінки критичності вразливостей:

– утиліта PT Exploit Explorer;

– Nessus Vulnerability Scanner.

Безкоштовна утиліта PT Exploit Explorer (PT EE) допомогає привернути увагу до уразливостей, які використовуються спеціальними хакерськими програмами, які були створені та опубліковані. Такий інструмент дозволяє будькому, навіть недосвідченому зловмиснику, використати вразливість і автоматично атакувати. PT Exploit Explorer дозволяє шукати посилання, що використовуються в загальнодоступних базах даних, включаючи Rapid7 та Exploit-db. Програма також забезпечує обробку текстових файлів, що містять список довільних вразливостей з бази даних CVE [31]. Скріншот роботи утиліти представлено на рис. 2.3.

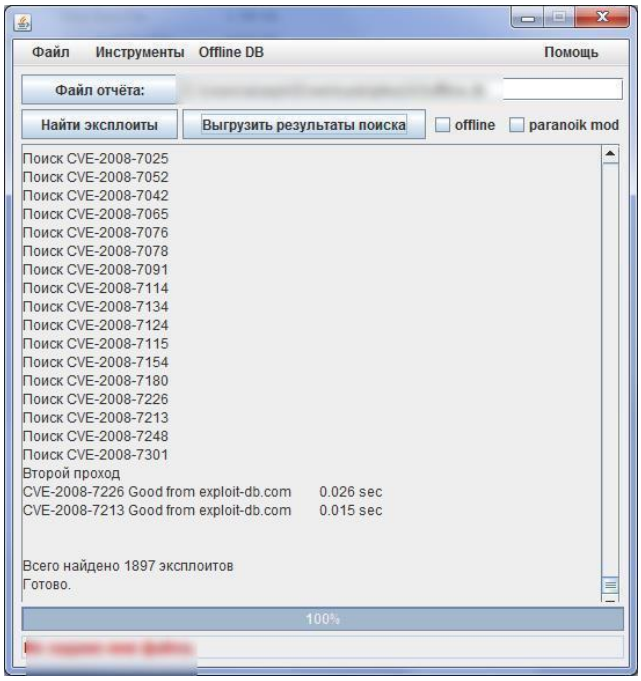

Рис. 2.3. Скріншот роботи утиліти

Nessus Vulnerability Scanner – один з найпопулярніших сканерів на вразливість. Це стандарт для сканерів на вразливість. Nessus складається з серверної та клієнтської частин. Сканер може сканувати кілька систем одночасно. Під час системного дослідження сканер запитає, чи є мережеві служби на всіх портах у межах діапазону налаштувань. Визначивши доступні послуги, він почав аналізувати його стійкість до крадіжок, імітуючи тип атаки, зазначений у модулі[32].

Виходячи з результатів сканування, маємо список з IP адресами та пов’язані з ними ризики. Ризики мають колірне кодування. Приклад роботи програми наведено на рис. 2.4.

У верхньому меню програми відображаються всі виявлені в мережі уразливості. Вибравши конкретну вразливість, можна отримати детальну інформацію про неї. Окрім опису вразливості, звіт може також містити пропозиції щодо його усунення. Як показано на малюнку. 2.4, Nessus здійснює оцінку за наступними категоріями: Info, Low, Medium, High. Крім того, сканер повідомляє кількість певних вразливостей за категоріями та створює діаграми для кращої візуалізації.

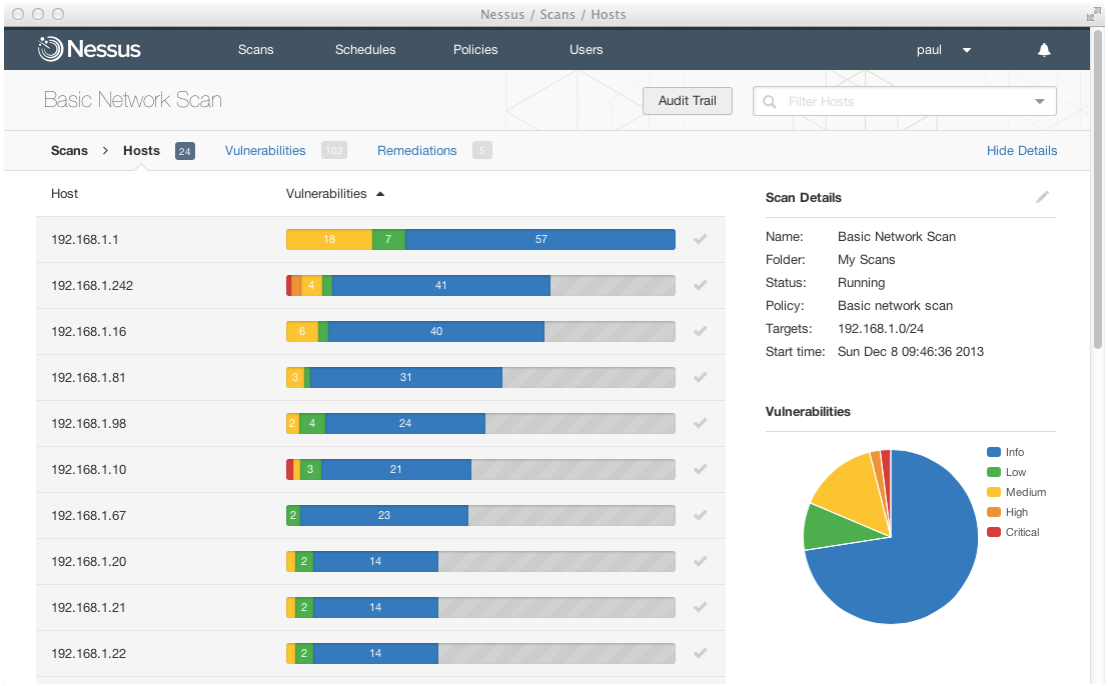

Рис. 2.4. Приклад роботи Nessus Vulnerability Scanner

РОЗДІЛ 3. РОЗРОБКА АЛГОРИТМУ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ ТА КОНТРОЛЮ

3.1. Вибір мови програмування

Для того, аби вирішити поставлене завдання, необхідна мова програмування, котра має відкриті бібліотека та являється об’єктно-орієнтовною. Нетехнічний критерій – наявність досвіду роботи з мовою. Під ці категорії потрапляють наступні мови програмування: C# та C++.

Найкраще, для досягання заданих цілей, підходить мова програмування C#, адже вона відповідає представленим критерієм. Для програмування на мові C# є необхідним середовище розробки, що має відладчик IntelliSense та аналізатор коду. Даним вимогам відповідає VisualStudio Community 2017, з оглядом на те, що розробка програмного забезпечення буде розроблена під керуванням операційної системи Windows 10.

3.2. Розробка алгоритму роботи програмного забезпечення

Алгоритм роботи програмного засобу, що здійснює оцінку ризику критичності вразливсоті представлений на рис. 3.1-3.3. На рис. 3.1. зображено перший етап роботи користувача з програмним забезпеченням. Користувач обирає параметри для здійснення пошуку: знайти всі вразливості для заданої операційної системи та кількість знайдених вразливсотей на сторінці результату.

На рис. 3.2. зображено другий етап - пошук вразливсоті, за допомогою її унікального ідентифікатора – CVE кодом. В кінці даного етапу, користувач отримує детальну інформацію, щодо вразливості та список продуктів, що містять її.

На рис. 3.3. зображено останній етап роботи програми – оцінка критичності виявленої вразливості.

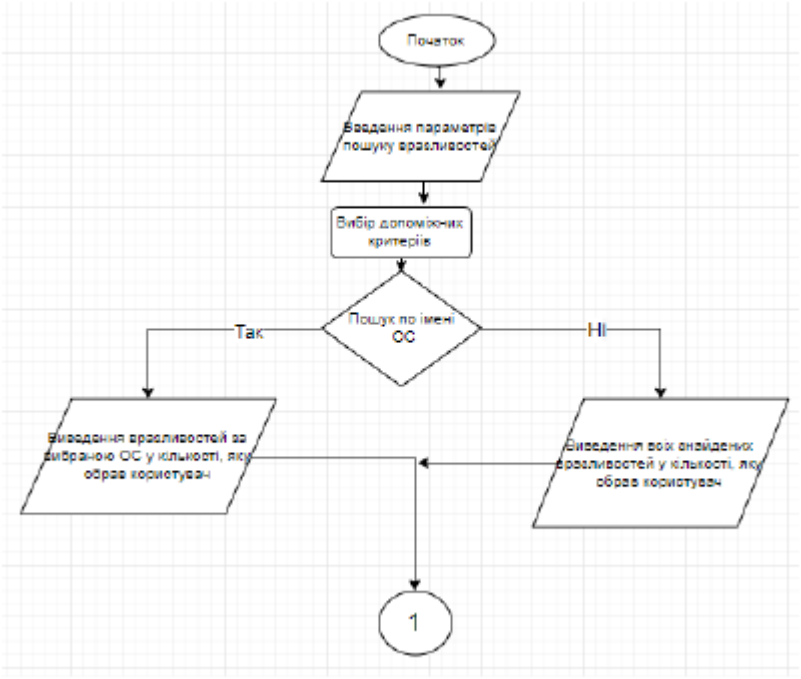

Рис. 3.1. Алгоритм першого етапу роботи програми

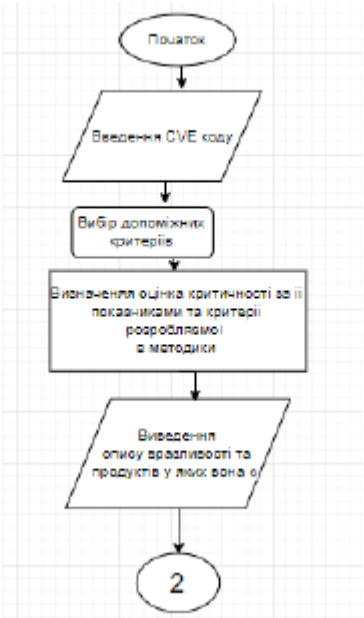

Рис. 3.2. Алгоритм другого етапу роботи програми

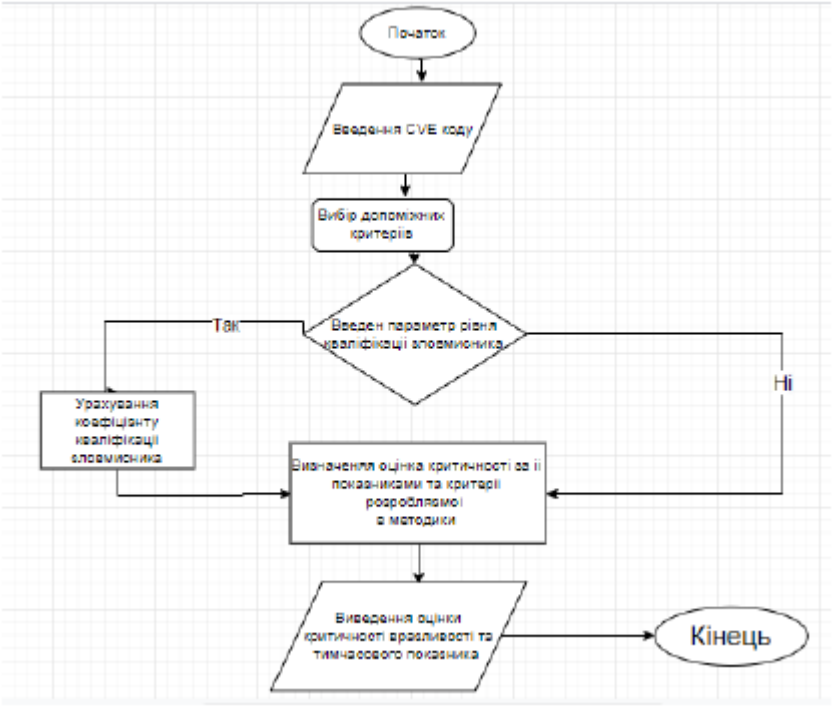

Рис. 3.3 Алгоритм останнього етапу роботи програми

3.3. Розробка інтерфейсу програми

Програма представляє з себе Application Programming Interface (API). На рис. 3.4 представлено зовнішній вигляд програми, за допомогою Swagger.

Рис. 3.4. Зовнішній вигляд

На рис. 3.5 представлено перелік точок прийому запиту на стороні серверу (endpoint).

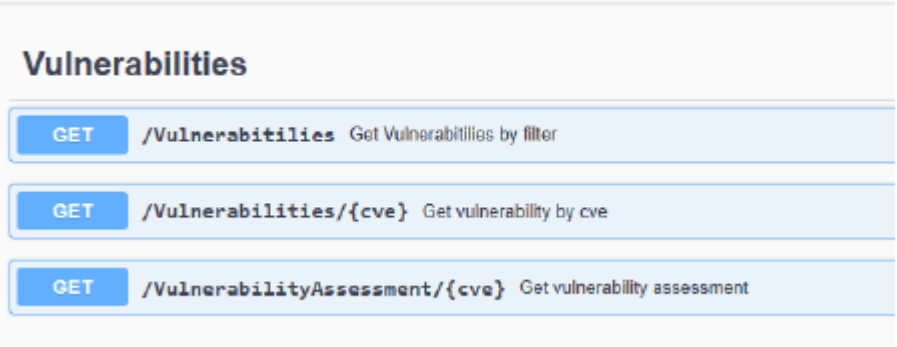

Рис. 3.5. Доступні точки прийому запиту на стороні серверу

3.4. Опис та тестування розробленого програмного забезпечення

Перша точка прийому запиту на стороні серверу – «/Vulnerabilities». Вона дає змогу користувачу отримати список вразливостей, за тими параметрами, що були введені користувачем. Приклад роботи цієї точки представлено на рис. 3.6-3.7.

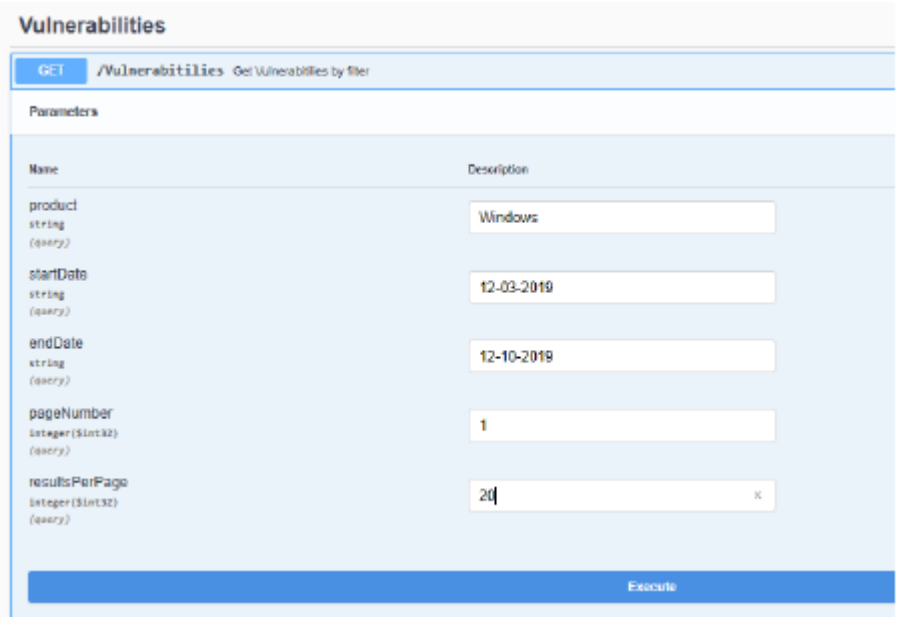

Рис. 3.6. Приклад вхідних даних для точки «/Vulnerabilities»

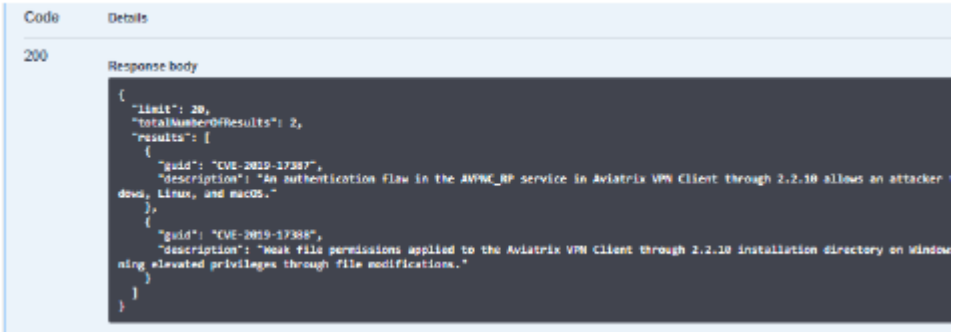

Рис. 3.7. Приклад вихідних даних для точки «/Vulnerabilities»

Друга точка – «/Vulnerability/{cve}». Вона дозволяє користувачу отримати детільну інформацію, щодо вразливості, за її унікальним ідентифікатором – CVE кодом. Приклад роботи даної точки, зображено на рис. 3.8-3.9.

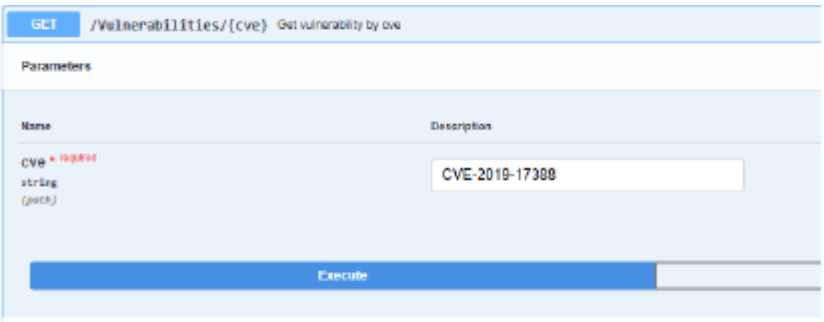

Рис. 3.8. Приклад вхідних даних для точки «/Vulnerability/{cve}»

Рис. 3.9. Приклад вихідних даних для точки «/Vulnerability/{cve}»

Остання точка – «/VulnerabilityAssessment/{cve}». Вона надає користувачу змогу отримати оцінку критичності вразливості, використовуючи при цьому її унікальний ідентифікатор – CVE код, та параметр, який охарактеризовує навики зловмисника. Приклад роботи зображено на рис. 3.10-3.11.

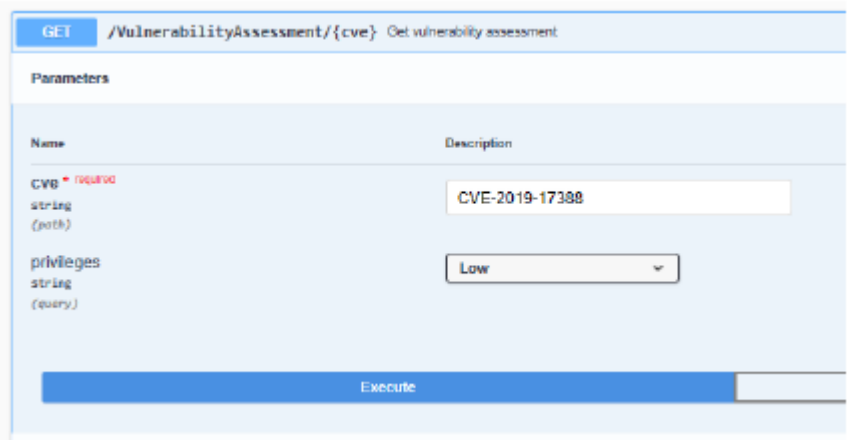

Рис. 3.10. Приклад вихідних даних для точки «/VulnerabilityAssessment/{cve}»

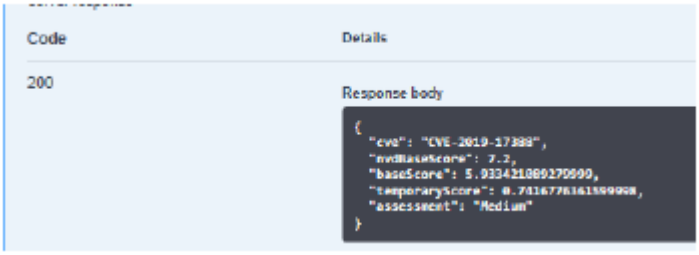

Рис. 3.11. Приклад вихідних даних для точки «/VulnerabilityAssessment/{cve}»

ВИСНОВКИ

У результаті виконання роботи вирішена задача створення програмного модуля моніторингу вразливостей GRID-систем.

Виконуючи роботу, були отримані наступні результати:

1. Проаналізовано принципи роботи Grid систем, існуючі вразливості, сучасні методи моніторингу та виявлення атак, а також відомі системи та програмні рішення моніторнигу вразливостей Grid систем, на основі чого, було зроблено висновок про необхідність розробки програмного модуля моніторингу вразливості Grid систем.

2.Розроблено програмний модуль, який забезпечує виконання моніторингу вразливостей GRID систем на основі загальної системи оцінки вразливостей (CVSS), що дозволяє здійснювати аналіз вразливостей системи для запобігання подальшого їх використання з метою порушення безпеки системи.

3. Проведено тестування розробленого програмного модулю моніторингу вразливостей Grid систем, для перевірки доцільності його використання, в якості одного з методів експертної характеристики вразливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Applying Network-Centric Approaches for Threat Detection and Response URL: https://www.gartner.com/en/documents/3904768/applying-network-centric-approaches-for-threat-detection

2. Распространенные угрозы ИБ в корпоративных сетях [URL: https://www.ptsecurity.com/ru-ru/research/analytics/network-traffic-analysis-2020/

3. Про захист інформації в інформаційно-телекомунікаційних системах [Текст]: Закон України № 80/94-ВР від 4 липня 2020 р. / Верховна Рада України // Відомості Верховної Ради України. – 1994. – №31. – Ст. 286.

4. Введение в информационную безопасность. Компьютеры: преступления, признаки уязвимости и меры защиты URL:http://www.bezpeka.com/ru/lib/sec/gen/art344.html

5. Лукацкий А. Обнаружение атак. — СПб.: БХВПетербург, 2001. – 624 с.

6. Офіційний сайт кіберполіції України URL:Режим доступу https:// https://cyberpolice.gov.ua/

7. М. А. Карпенко, студентка; и О. В. Коломієць, студент, Аналіз сучасних систем виявленя вторгнень в іот та запобіганя їм URL:http://www.itrew.ru/windows/statistika-operacionnykh-sistem-za-ap.html.

8. Технологии обнаружения URL:https://www.bytemag.ru/articles/detail.php?ID=6850&sphrase_id=38331

9. J.P. Anderson, Computer Security Threat Monitoring and Surveillance // James P. Anderson Co., Fort Washington, PA, April. 1980, P. 75-127.

10. Городецкий В.И., Котенко И.В., Карсаев О.В., Хабаров А.В. Многоагентные технологии комплексной защиты информации в телекоммуникационных системах. ISINAS. 2000. – №3. – P. 89-92.

11. Бараматова И. С. Зайцева Е. В. Состояние и перспективы развития систем обнаружения компьютерных вторжений // Горный информационно-аналитический бюллетень (научно-технический журнал). Вып. S6. 2011

12. Кількість мобільних пристроїв з інтернетом скоро перевищить населення Землі URL: http://fmf.udpu.org.ua/novyny-suchasnoinauky/593-kilkist-mobilnykh-prystroiv-na-zemliperevyshchyla-kilkist-liudei.

13. Aberrant Behavior Detection in Time Series for Monitoring Business-Critical Metrics URL:http://www.imvu.com/technology/anomalous-behavior.pdf

14. Anomaly detection for monitoring: A statistical approach to time series anomaly detection. O'Reilly. URL: http://www.oreilly.com/webops-perf/free/files/anomaly-detection-monitoring.pdf

15. Context-Aware Time Series Anomaly Detection for Complex Systems. Proceedings of the SDM Workshop. URL: https://www.microsoft.com/en-us/research/publication/context-aware-time-series-anomaly-detection-for-complex-systems/

16. Long Short Term Memory Networks for Anomaly Detection in Time Series. ESANN 2015 Proceedings. URL: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2015-56.pdf

17. S. Agrawal, J. Agrawal, "Survey on Anomaly Detection using Data Mining Techniques", Procedia Computer Science, vol. 60, 2015, pp. 708 – 713.

18. C. Chio, D. Freeman. "Machine Learning and Security", O'Reilly Media, Inc., – URL: https://github.com/oreilly-mlsec/bookresources/tree/master/chapter3/datasets/cpu-utilization. – 01.12.2017.

19. Network Monitoring Best Practices [Електроний ресурс] – URL: https://www.dnsstuff.com/how-to-monitor-network-traffic

20. Dover D., Dafforn E. Search Engine Optimization Secrets. Indianapolis: Wiley Publishing, Inc., 2011. 456 p.

21. Cardoso, J., Sheth, Amit (Eds.) ,. Semantic Neural Services, Processes and Applications. - Springer, 2006.

22. Томас Коннолли. Базы данных. Проектирование, реализация и сопровождение. Теория и практика.

23. Роберт К. Элсенпитер, Тоби Дж. Велт. Нейронные сети строим сами. 2006. - 256 с.

24. WiMAX Forum URL:wimaxforum.org/

25. WLAN: практическое руководство для администраторов и профессиональных пользователей» / Томас Мауфер. – М.: КУДИЦ-Образ, 2005

26. Марк Лутц. Программирование на Python / Пер. с англ. - Четвѐртый изд. - СПб .: Символ-Плюс, 2011. - Т. I. - 992 с.

27. Andersson, O. Threat, risk, and vulnerability analyses during the development of IT systems in the Swedish Armed Forces./ Ola Andersson – Sweden: Umeå University Department of Computing Science SE-901 87 UMEÅ SWEDEN.

28. Полное руководство по общему стандарту оценки уязвимостей [Електроний ресурс] // securitylab – 31.05.2019. Режим доступу: https://www.securitylab.ru/analytics/355336.php;

29. Системы оценки уязвимостей [Електроний ресурс] // Windows IT Pro/RE – 31.05.2019. Режим доступу: https://www.osp.ru/winitpro/2006/02/1156304/;

30. Сканеры уязвимостей [Електроний ресурс] // it-black – 31.05.2019. Режим доступу: https://it-black.ru/skanery-uyazvimostey/;

31. Системы оценки уязвимостей [Електроний ресурс] // Windows IT Pro/RE – 31.05.2019. Режим доступу: https://www.osp.ru/winitpro/2006/02/1156304/;

32. About OSVDB [Електроний ресурс] // blog.osvdb – 30.05.2019. Режим доступу: https://blog.osvdb.org/about/.

ДОДАТОК А

Код програмного продукту

```
namespace NVDHunter.Controllers
{
    [ApiController]
    public class VulnerabilitiesController : ControllerBase
    {
        private readonly IVulnerabilitiesManager vulnerabilitiesManager;
        /// <summary>
        /// Creates instance of a controller
        /// </summary>
        /// <param name="vulnerabilitiesManager"></param>
        public VulnerabilitiesController(IVulnerabilitiesManager vulnerabilitiesManager)
        {
            this.vulnerabilitiesManager = vulnerabilitiesManager;
        }
        /// <summary>
        /// Get Vulnerabitilies by filter
        /// </summary>
        ///
        /// <returns></returns>
         [HttpGet("Vulnerabitilies")]
         [ProducesResponseType(200)]
         [ProducesResponseType(404)]
        public ActionResult<Vulnerabilities> Get(string product, [FromQuery] Filter filter)
        {
            try
            {
```

```
                var response = this.vulnerabilitiesManager.Get(product, filter);
                return this.Ok(response);
            }
            catch (Exception ex)
            {
                return this.BadRequest(ex.Message);
            }
        }
        /// <summary>
        /// Get vulnerability by cve
        /// </summary>
        /// <param name="cve"></param>
        /// <returns></returns>
         [HttpGet("Vulnerabilities/{cve}")]
         [ProducesResponseType(200)]
         [ProducesResponseType(400)]
         [ProducesResponseType(404)]
        public ActionResult<VulnerabilityDetails> GetById(string cve)
        {
            try
            {
                var response = this.vulnerabilitiesManager.GetById(cve);
                if (response == null)
                {
                    return this.NotFound();
                }
                return this.Ok(response);
            }
            catch (Exception ex)
```

```
            {
                return this.BadRequest(ex.Message);
            }
        }
        /// <summary>
        /// Get vulnerability assessment
        /// </summary>
        /// <param name="cve"></param>
        /// <param name="privileges"></param>
        /// <returns></returns>
         [HttpGet("VulnerabilityAssessment/{cve}")]
         [ProducesResponseType(200)]
         [ProducesResponseType(400)]
         [ProducesResponseType(404)]
        public ActionResult<VulnerabilityAssessment> GetAssessment(string
cve, [FromQuery] PrivilegesEnum privileges)
        {
            try
            {
                string privilegesValue = null;
                if ((int)privileges == 0)
                {
                    privilegesValue = "L";
                }
                else if ((int)privileges == 1)
                {
                    privilegesValue = "H";
                }
                else
                {
```

```
                    privilegesValue = "N";
                }
                var response = this.vulnerabilitiesManager.GetAssessment(cve, privilegesValue);
                if (response == null)
                {
                    return this.NotFound();
                }
                return this.Ok(response);
            }
            catch (Exception ex)
            {
                return this.BadRequest(ex.Message);
            }
        }
    }
    }
namespace NVDHunter.API.Extensions.Swagger
{
    public static class SwaggerConfiguration
    {
        public static void SwaggerConfigureServices(this IServiceCollection services, IConfiguration configuration, Info swaggerInfo = null)
        {
            var appName = "NVDHunter";
            if (swaggerInfo == null)
            {
                swaggerInfo = SwaggerConstants.GetDefaultInfo(appName);
            }
            services.AddSwaggerGen(conf =>
```

```
                {
                    conf.SwaggerDoc("swagger", swaggerInfo);
                    conf.IncludeXmlComments(GetXmlFilePath(appName));
        //conf.IncludeXmlComments(Path.Combine(Path.GetDirectoryName(Assembl
y.GetEntryAssembly().Location), "NVDHunter.xml"));
                    conf.ResolveConflictingActions(apiDescriptions => apiDe-
scriptions.First());
                    conf.CustomOperationIds(x                              =>
$"{x.HttpMethod}_{x.RelativePath}");
                    var security = new Dictionary<string, IEnumerable<string>>
                    {
                        {"cookieAuth", Enumerable.Empty<string>()},
                    };
                    conf.AddSecurityRequirement(security);
                    conf.DescribeAllEnumsAsStrings();
                });
            }
            public static void UseCustomSwagger(this IApplicationBuilder app)
            {
                app.UseSwagger();
                app.UseSwaggerUI(conf =>
                {
                    conf.SwaggerEndpoint($"swagger/swagger.json",
string.Empty);
                    conf.DocExpansion(DocExpansion.None);
                });
            }
            private static string GetXmlFilePath(string fileName)
            {
                var xmlFile = $"{fileName}.xml";
```

```
                return Path.Combine(AppContext.BaseDirectory, xmlFile);
            }
        }
    }
    namespace NVDHunter.API.Extensions.Swagger
    {
        public static class SwaggerConstants
        {
            public static Info GetDefaultInfo(string appName)
            {
                return new Info
                {
                    Title = appName,
                    Description = "Vulnerabilities assessment API"
                };
             }
        }
    }
    namespace NVDHunter.BussinessLogic.CountersFunctionality
    {
        public static class AssessmentGenerator
        {
            public static string GetAssessment(double baseScore)
            {
                if (baseScore <= AssessmentsConstants.MAX_LOW_VALUE)
                {
                    return AssessmentsConstants.LOW;
                }
```

```
                    if (baseScore >= AssessmentsConstants.MIN_MEDIUM_VALUE && baseScore <= AssessmentsConstants.MAX_MEDIUM_VALUE)
                    {
                        return AssessmentsConstants.MEDIUM;
                    }
                    if (baseScore >= AssessmentsConstants.MIN_HIGH_VALUE && baseScore <= AssessmentsConstants.MAX_HIGH_VALUE)
                    {
                        return AssessmentsConstants.HIGH;
                    }
                    return AssessmentsConstants.CRITICAL;
                }
            }
        }
        namespace NVDHunter.BussinessLogic.CountersFunctionality
        {
            public class BaseScoreCounter
            {
                private readonly Dictionary<string, string> marksDictionary;
                public BaseScoreCounter(string vectorString)
                {
                    this.marksDictionary = MarksDictionaryGenerator.CreateMarksDictionary(vectorString);
                }
                public double GetBaseScore(string privilegesString)
                {
                    var baseScore = this.CalculateAuxiliaryValues(privilegesString);
                    return baseScore;
```

```
        }
        private double CalculateAuxiliaryValues(string privilegesString)
        {
            var baseScore = 0.0;
            var av = this.GetAttactVectorScore();
            var ac = this.GetAttactComplexityScore();
            var ui = this.GetUserInteractionScore();
            var ci = this.GetConfidentialityImpactScore();
            var ii = this.GetIntegrityImpactScore();
            var ai = this.GetAvailabilityImpactScore();
            var pr = this.GetPrivilegesRequiredScore(privilegesString);
            var exploitability = 20 * av * ac * pr * ui;
            var preISC = 10.41*(1 - (1 - ci) * (1 - ii) * (1 - ai));
            baseScore = this.GenerateBaseScore(preISC, exploitability);
            return baseScore;
        }
        private double GetAttactVectorScore()
        {
            string paramMark = null;
            if
(this.marksDictionary.TryGetValue(VectorConstants.ATTACK_VECTOR, out pa-
ramMark))
            {
                if (paramMark == Marks-
Conctants.ADJACENT_NETWORK)
                {
                    return ValuesOf-
Marks.AV_ADJACENT_NETWORK_VALUES;
                }
                if (paramMark == MarksConctants.NETWORK)
```

```
                {
                    return ValuesOfMarks.AV_NETWORK_VALUES;
                }
                if (paramMark == MarksConctants.LOCAL)
                {
                    return ValuesOfMarks.AV_LOCAL_VALUES;

                if (paramMark == MarksConctants.PHYSICAL)
                {
                    return ValuesOfMarks.AV_PHYSICAL_VALUES;
                }
            }
            return ValuesOfMarks.AV_NONE_VALUES;
        }
        private double GetAttactComplexityScore()
        {
            string paramMark = null;
            if
(this.marksDictionary.TryGetValue(VectorConstants.ATTACK_COMPLEXITY, out
paramMark))
            {
                if (paramMark == MarksConctants.LOW)
                {
                    return ValuesOfMarks.AC_LOW_VALUES;
                }
                if (paramMark == MarksConctants.HIGH)
                {
                    return ValuesOfMarks.AC_HIGH_VALUES;
                }
            }
```

```
            return ValuesOfMarks.AC_NONE_VALUES;
        }
        private double GetPrivilegesRequiredScore(string privileg-
esString)
        {
            if (privilegesString == MarksConctants.LOW)
            {
                return ValuesOfMarks.PR_LOW_VALUES;
            }
            if (privilegesString == MarksConctants.HIGH)
            {
                return ValuesOfMarks.PR_HIGH_VALUES;
            }
            return ValuesOfMarks.PR_NONE_VALUES;
        }
        private double GetUserInteractionScore()
        {
            string paramMark = null;
            if
(this.marksDictionary.TryGetValue(VectorConstants.USER_INTERACTION,  out
paramMark))
            {
                if (paramMark == MarksConctants.REQUIRED)
                {
                    return ValuesOfMarks.UI_REQUIRED_VALUES;
                }
                if (paramMark == MarksConctants.NONE)
                {
                    return ValuesOfMarks.UI_NONE_VALUES;
                }
```

```
            }
            return ValuesOfMarks.UI_NOT_DEFINED_VALUES;
        }
        private double GenerateBaseScore(double preISC, double exploitability)
        {
            string paramMark = null;
            double authScore = 0.0;
            double isc = 0.0;
            if (this.marksDictionary.TryGetValue(VectorConstants.SCOPE, out paramMark))
            {
                if (paramMark == MarksConctants.CHANGED)
                {
                    authScore = ValuesOfMarks.S_CHANGED_VALUES;
                    isc = authScore * (preISC - 0.029) - 3.25 * Math.Pow((preISC - 0.02), 15.0);
                    return Math.Abs(1.08 * (6*isc + 4 * exploitability - 1.5) * authScore);
                }
                if (paramMark == MarksConctants.UNCHANGED)
                {
                    authScore = ValuesOfMarks.S_UNCHANGED_VALUES;
                    isc = 1.176 + authScore;
                    return Math.Abs(1.08 * (0.6 * preISC + 0.4 * exploitability - 1.5) * isc);
                }
            }
```

```
authScore = ValuesOfMarks.S_NONE_VALUES;
isc = 1.176 + authScore;
var test = 0.6 * preISC;
var t2 = 0.4 * exploitability + 1.5;
var t3 = 1.08 * isc;
var t4 = t3 * (test + t2);
var t5 = Math.Abs(t4);
return t5;
}
private double GetConfidentialityImpactScore()
{
    string paramMark = null;
    if
(this.marksDictionary.TryGetValue(VectorConstants.CONFIDENTIALITY_IMPAC
T, out paramMark))
    {
        if (paramMark == MarksConctants.LOW)
        {
            return ValuesOfMarks.CI_LOW_VALUES;
        }
        if (paramMark == MarksConctants.HIGH)
        {
            return ValuesOfMarks.CI_HIGH_VALUES;
        }
        if (paramMark == MarksConctants.NONE)

            return ValuesOfMarks.CI_NONE_VALUES;
        }
    }
    return ValuesOfMarks.CI_NOT_DEFINED_VALUES;
```

```
                }
                private double GetIntegrityImpactScore()
                {
                    string paramMark = null;
                    if (marksDiction-
ary.TryGetValue(VectorConstants.INTEGRITY_IMPACT, out paramMark))
                    {
                        if (paramMark == MarksConctants.LOW)
                        {
                            return ValuesOfMarks.I_LOW_VALUES;
                        }
                        if (paramMark == MarksConctants.HIGH)
                        {
                            return ValuesOfMarks.I_HIGH_VALUES;
                        }
                        if (paramMark == MarksConctants.NONE)
                        {
                            return ValuesOfMarks.I_NONE_VALUES;
                        }
                    }
                    return ValuesOfMarks.I_NOT_DEFINED_VALUES;
                }
                private double GetAvailabilityImpactScore()
                {
                    string paramMark = null;
                    if
(this.marksDictionary.TryGetValue(VectorConstants.AVAILABILITY_IMPACT,
out paramMark))
                    {
                        if (paramMark == MarksConctants.LOW)
```

```
                {
                    return ValuesOfMarks.A_LOW_VALUES;
                }
                if (paramMark == MarksConctants.HIGH)
                {
                    return ValuesOfMarks.A_HIGH_VALUES;
                }
                if (paramMark == MarksConctants.NONE)
                {
                    return ValuesOfMarks.A_NONE_VALUES;
                }
            }
            return ValuesOfMarks.A_NOT_DEFINED_VALUES;
        }
    }
}
namespace NVDHunter.BussinessLogic.CountersFunctionality
{
    public static class MarksDictionaryGenerator
    {
        public static Dictionary<string, string> CreateMarksDictionary(string vectorString)
        {
            vectorString = vectorString.ToUpper();
            var marksKeyValuePairs = new Dictionary<string, string>();
            var arrayMarks = vectorString.Split(new string[] { "/" }, String-SplitOptions.RemoveEmptyEntries);
            foreach (var item in arrayMarks)
            {
```

```
                    var keyValueArray = item.Split(new string[] { ":" }, String-
SplitOptions.RemoveEmptyEntries);
                    marksKeyValuePairs.Add(keyValueArray[0],  keyValueAr-
ray[1]);
                }
                return marksKeyValuePairs;
            }
        }
    }
    namespace NVDHunter.BussinessLogic.CountersFunctionality
    {
        public class TemporalyScoreCounter
        {
            private readonly Dictionary<string, string> marksDictionary;
            public TemporalyScoreCounter(string vectorString)
            {
                this.marksDictionary =
    MarksDictionaryGenerator.CreateMarksDictionary(vectorString);
            }
            public double GetTemporalyScore(double baseScore)
            {
                double temporalyScore = 0.0;
                var exploitability = this.GenereteExploitabilityScore();
                var remendiationLevel = this.GenereteRemediationLevelScore();
                var report = this.GenereteReportScore();
                temporalyScore = baseScore * exploitability * remendiationLevel
* report;
                return temporalyScore;
            }
            private double GenereteExploitabilityScore()
```

```
        {
            string paramMark = null;
            if
(this.marksDictionary.TryGetValue(VectorConstants.EXPLOITABILITY, out pa-
ramMark))
            {
                if (paramMark == MarksConctants.EXPLOIT_EXIST)
                {
                    return                                    ValuesOf-
Marks.EXPLOITABILITY_UNPROVEN_VALUES;
                }
                if              (paramMark          ==          Marks-
Conctants.FUNCTIONAL_EXPLOIT_EXIST)
                {
                    return                                    ValuesOf-
Marks.EXPLOITABILITY_FUNCTIONAL_VALUES;
                }
                if              (paramMark          ==          Marks-
Conctants.PROOF_OF_CONCEPT_CODE)
                {
                    return                                    ValuesOf-
Marks.EXPLOITABILITY_PROOF_OF_CONCERT_VALUES;
                }
                if (paramMark == MarksConctants.HIGH)
                {
                    return                                    ValuesOf-
Marks.EXPLOITABILITY_HIGH_VALUES;
                }
                if (paramMark == MarksConctants.NOT_DEFINED)
                {
```

```
                    return ValuesOf-
Marks.EXPLOITABILITY_NOT_DEFINED_VALUES;
                }
            }
            return ValuesOfMarks.EXPLOITABILITY_NONE_VALUES;
        }
        private double GenereteRemediationLevelScore()
        {
            string paramMark = null;
            if
(this.marksDictionary.TryGetValue(VectorConstants.REMEDIATION_LEVEL, out
paramMark))
            {
                if (paramMark == MarksConctants.OFFICIAL_FIX)
                {
                    return ValuesOf-
Marks.REMEDIATION_OFFICIAL_FIX_VALUES;
                }
                if (paramMark == MarksConctants.TEMPORARY_FIX)
                {
                    return ValuesOf-
Marks.REMEDIATION_TEMPORARY_FIX_VALUES;
                }
                if (paramMark == MarksConctants.WORKAROUND)
                {
                    return ValuesOf-
Marks.REMEDIATION_WORKAROUND_VALUES;
                }
                if (paramMark == MarksConctants.UNAVAILABLE)
                {
```

```
                    return ValuesOf-
Marks.REMEDIATION_UNAVAILABLE_VALUES;
                }
                if (paramMark == MarksConctants.NOT_DEFINED)
                {
                    return ValuesOf-
Marks.REMEDIATION_NOT_DEFINED_VALUES;
                }
            }
            return ValuesOfMarks.REMEDIATION_NONE_VALUES;
        }
        private double GenereteReportScore()
        {
            string paramMark = null;
            if
(this.marksDictionary.TryGetValue(VectorConstants.REPORT_CONFIDENCE, out
paramMark))
            {
                if (paramMark == MarksConctants.UNKNOWN)
                {
                    return ValuesOf-
Marks.REPORT_UNCONFIRMED_VALUES;
                }
                if (paramMark == MarksConctants.CONFIRMED)
                {
                    return ValuesOf-
Marks.REPORT_CONFIRMED_VALUES;
                }
                if (paramMark == MarksConctants.REASONABLE)
                {
```

```
                    return ValuesOf-
Marks.REPORT_UNCORROBORATED_VALUES;
                }
                if (paramMark == MarksConctants.NOT_DEFINED)
                {
                    return ValuesOf-
Marks.REPORT_NOT_DEFINED_VALUES;
                }
            }
            return ValuesOfMarks.REPORT_NONE_VALUES;
        }
    }
}
namespace NVDHunter.BussinessLogic
{
    public interface IVulnerabilitiesManager
    {
        Vulnerabilities Get(string product, Filter filter);
        VulnerabilityDetails GetById(string cve);
        VulnerabilityAssessment GetAssessment(string cve, string privileg-
esString);
    }
}
namespace NVDHunter.BussinessLogic
{
    public class VulnerabilitiesManager : IVulnerabilitiesManager
    {
        private RestClient client;
        public VulnerabilitiesManager()
        {
```

```
            this.client = new RestClient();
        }
        public Vulnerabilities Get(string product, Filter filter)
        {
            this.client = new RestClient();
            this.client.BaseUrl = new Uri(UrlConstants.GET_ALL);
            var content = filter.GetType().GetProperties(BindingFlags.Instance | BindingFlags.Public)
            .ToDictionary(prop => prop.Name, prop => prop.GetValue(filter, null));
            var response = this.Send(Method.GET, OtherConstants.LIST, content);
            var result = JsonConvert.DeserializeObject<Vulnerabilities>(response.Content);
            if (product != null)
            {
                var resultsByProduct = result.Results.Where(x => x.Description.Contains(product)).ToList();
                result.Results = resultsByProduct;
                result.TotalNumberOfResults = resultsByProduct.Count;
            }
            return result;
        }
        public VulnerabilityDetails GetById(string cve)
        {
            this.client.BaseUrl = new Uri(UrlConstants.GET_BY_ID_URL);
            var response = this.Send(Method.GET, cve + OtherConstants.JSON);
            if (response.StatusCode == HttpStatusCode.NotFound)
            {
```

```
                return null;
            }
            var result = this.MapToDetailsModel(response);
            return result;
        }
        public VulnerabilityAssessment GetAssessment(string cve, string privilegesString)
        {
            var vulnerabilityDetails = this.GetById(cve);
            var baseScore = new BaseScoreCounter(vulnerabilityDetails.VectorString);
            var temporalyScore = new TemporalyScoreCounter(vulnerabilityDetails.VectorString);
            var baseScoreValue = baseScore.GetBaseScore(privilegesString);
            var temporaryScoreValue = temporalyScore.GetTemporalyScore(baseScoreValue);
            var assessment = AssessmentGenerator.GetAssessment(baseScoreValue);
            return new VulnerabilityAssessment()
            {
                CVE = vulnerabilityDetails.Guid,
                NVDBaseScore = vulnerabilityDetails.NVDBaseScore,
                BaseScore = baseScoreValue,
                TemporaryScore = temporaryScoreValue,
                Assessment = assessment
            };
        }
        private VulnerabilityDetails MapToDetailsModel(IRestResponse response)
        {
```

```
                var details = new VulnerabilityDetails();
                var jObject = JObject.Parse(response.Content);
                details.Description =
        jObject.SelectToken("cve.description.description_data.[0].value").ToString();
                details.Guid =
        jObject.SelectToken("cve.CVE_data_meta.ID").ToString();
                details.Vendors =
        jObject.SelectToken("cve.affects.vendor.vendor_data")?.Select(x => new Ven-
dor()
                {
                    VendorName = (string)x.SelectToken("vendor_name"),
                    Products = x.SelectToken("product.product_data")?.Select(p
=> new Product()
                    {
                        ProductName = (string)p.SelectToken("product_name"),
                        Versions =
p.SelectToken("version.version_data")?.Select(v =>
(string)v.SelectToken("version_value")).ToList()
                    }).ToList()
                }).ToList();
                details.VectorString =
        (string)jObject.SelectToken("impact.baseMetricV3.cvssV3.vectorString") ??
        (string)jObject.SelectToken("impact.baseMetricV2.cvssV2.vectorString");
                details.NVDBaseScore =
        (double?)jObject.SelectToken("impact.baseMetricV3.cvssV3.baseScore") ??
        (double?)jObject.SelectToken("impact.baseMetricV2.cvssV2.baseScore");
                return details;
            }
            private IRestResponse Send(Method method, string query, Diction-
ary<string, object> content = null)
```

```
        {
            var httpRequest = new RestRequest(query, method);
            if (content != null)
            {
                httpRequest = this.AddRarametersToRequest(httpRequest, content);
            }
            var response = this.client.Execute(httpRequest);
            if (response.StatusCode == HttpStatusCode.Unauthorized)
            {
                this.client.AddDefaultHeader(Constants.AUTH, Constants.TOKEN);
                response = this.Send(method, query, content);
            }
            return response;
        }
        private RestRequest AddRarametersToRequest(RestRequest request, Dictionary<string, object> content)
        {
            foreach (var keyValue in content)
            {
                request.AddParameter(keyValue.Key, keyValue.Value);
            }
            return request;
        }
    namespace NVDHunter.Core.Constants.Metrics
    {
        public static class MarksConctants
        {
            //ATTACK VECTOR MARKS
```

```
        public const string NETWORK = "N";
        public const string ADJACENT_NETWORK = "A";
        public const string LOCAL = "L";
        public const string PHYSICAL = "P";
        //COMMON MARKS
        public const string NONE = "N";
        public const string LOW = "L";
        public const string HIGH = "H";
        public const string MEDIUM = "M";
        public const string NOT_DEFINED = "X";
        //USER INTERACTION MARKS
        public const string REQUIRED = "R";
        //SCOPE MARKS
        public const string UNCHANGED = "U";
        public const string CHANGED = "C";
        //EXPLOITABILITY MARKS
        public const string EXPLOIT_EXIST = "U";
        public const string PROOF_OF_CONCEPT_CODE= "P";
        public const string FUNCTIONAL_EXPLOIT_EXIST = "F";
        //REMEDIATION LEVEL MARKS
        public const string OFFICIAL_FIX = "O";
        public const string TEMPORARY_FIX = "T";
        public const string WORKAROUND = "W";
        public const string UNAVAILABLE = "U";
        //REPORT CONFIDENCE MARKS
        public const string UNKNOWN = "U";
        public const string REASONABLE = "R";
        public const string CONFIRMED = "C";
    }
}
```

```
public static class ValuesOfMarks
    {
        //ATTACK VECTOR MARKS VALUES
        public const double AV_NONE_VALUES = 0.2;
        public const double AV_ADJACENT_NETWORK_VALUES = 0.9;
        public const double AV_LOCAL_VALUES = 1.0;
        public const double AV_NETWORK_VALUES = 0.8;
        public const double AV_PHYSICAL_VALUES = 1.5;
        //ATTACK COMPLEXITY MARKS VALUES
        public const double AC_LOW_VALUES = 1;
        public const double AC_HIGH_VALUES = 1.5;
        public const double AC_NONE_VALUES = 0.2;
        //PRIVILEGES REQUIRED MARKS VALUES
        public const double PR_LOW_VALUES = 1;
        public const double PR_HIGH_VALUES = 2;
        public const double PR_NONE_VALUES = 0.2;
        //USER INTERACTION MARKS VALUES
        public const double UI_REQUIRED_VALUES = 0.3;
        public const double UI_NONE_VALUES = 0.5;
        public const double UI_NOT_DEFINED_VALUES = 0.1;
        //SCOPE MARKS VALUES
        public const double S_UNCHANGED_VALUES = 0.7;
        public const double S_CHANGED_VALUES = 1;
        public const double S_NONE_VALUES = 0.2;
        //CONFIDENTIALITY IMPACT MARKS VALUES
        public const double CI_LOW_VALUES = 1.0;
        public const double CI_HIGH_VALUES = 1.5;
        public const double CI_NONE_VALUES = 0.4;
        public const double CI_NOT_DEFINED_VALUES = 0.1;
        //INTEGRITY IMPACT MARKS VALUES
```

```
public const double I_LOW_VALUES = 1.0;
public const double I_HIGH_VALUES = 1.5;
public const double I_NONE_VALUES = 0.4;
public const double I_NOT_DEFINED_VALUES = 0.1;
//AVAILABILITY IMPACT MARKS VALUES
public const double A_LOW_VALUES = 1.0;
public const double A_HIGH_VALUES = 2.0;
public const double A_NONE_VALUES = 0.4;
public const double A_NOT_DEFINED_VALUES = 0.1;
//EXPLOITABILITY MARKS VALUES
public const double EXPLOITABILITY_UNPROVEN_VALUES =
0.85;
public const double EXPLOITABIL-
ITY_PROOF_OF_CONCERT_VALUES = 0.9;
public const double EXPLOITABILITY_FUNCTIONAL_VALUES
= 0.95;
public const double EXPLOITABILITY_HIGH_VALUES = 1.0;
public const double EXPLOITABILITY_NOT_DEFINED_VALUES
= 1.0;
public const double EXPLOITABILITY_NONE_VALUES = 0.5;
//REMEDIATION LEVEL MARKS VALUES
public const double REMEDIATION_OFFICIAL_FIX_VALUES =
0.87;
public const double REMEDIATION_TEMPORARY_FIX_VALUES
= 0.9;
public const double REMEDIATION_WORKAROUND_VALUES =
0.95;
public const double REMEDIATION_UNAVAILABLE_VALUES =
1.0;
```

```
            public const double REMEDIATION_NOT_DEFINED_VALUES =
1.0;
            public const double REMEDIATION_NONE_VALUES = 0.5;
            //REPORT CONFIDENCE MARKS VALUES
            public const double REPORT_NOT_DEFINED_VALUES = 1.0;
            public const double REPORT_NONE_VALUES = 0.5;
            public const double REPORT_UNCONFIRMED_VALUES = 0.9;
            public const double REPORT_UNCORROBORATED_VALUES =
0.95;
            public const double REPORT_CONFIRMED_VALUES = 1.0;
        }
    }
    namespace NVDHunter.Core.Constants.Metrics
    {
        public static class VectorConstants
        {
            //BASE METRICS
            public const string ATTACK_VECTOR = "AV";
            public const string ATTACK_COMPLEXITY = "AC";
            public const string PRIVILEGES_REQUIRED = "PR";
            public const string USER_INTERACTION = "UI";
            public const string SCOPE = "S";
            public const string CONFIDENTIALITY_IMPACT = "C";
            public const string INTEGRITY_IMPACT = "I";
            public const string AVAILABILITY_IMPACT = "A";
            //TEMPORAL METRICS
            public const string EXPLOITABILITY = "E";
            public const string REMEDIATION_LEVEL = "RL";
            public const string REPORT_CONFIDENCE = "RC";
            //ENVIRONMENTAL METRICS
```

```
            public const string CONFIDENTIALITY_REQUIREMENT = "CR";
            public const string INTEGRITY_REQUIREMENT = "IR";
            public const string AVAILABILITY_REQUIREMENT = "AR";
        }
    }
    namespace NVDHunter.Core.Constants
    {
        public static class AssessmentsConstants
        {
            //Assessment names
            public const string LOW = "Low";
            public const string MEDIUM = "Medium";
            public const string HIGH = "High";
            public const string CRITICAL = "Critical";
            //Assessment values
            public const double MIN_LOW_VALUE = 0.1;
            public const double MAX_LOW_VALUE = 3.9;
            public const double MIN_MEDIUM_VALUE = 4.0;
            public const double MAX_MEDIUM_VALUE = 6.9;
            public const double MIN_HIGH_VALUE = 7.0;
            public const double MAX_HIGH_VALUE = 9.4;
            public const double MIN_CRITICAL_VALUE = 9.5;
            public const double MAX_CRITICAL_VALUE = 10.0;
        }
    }
    namespace NVDHunter.Core.Enums
    {
        public enum PrivilegesEnum
        {
            Low,
```

```
            High
        }
    }
    namespace NVDHunter.Core.Models.ResponseOfDetails
    {
        public class Product
        {
            public string ProductName { get; set; }
            public List<string> Versions { get; set; } = new List<string>();
        }
    }
    namespace NVDHunter.Core.Models.ResponseOfDetails
    {
        public class Vendor
        {
            public string VendorName { get; set; }
            public List<Product> Products { get; set; } = new List<Product>();
        }
    }
    namespace NVDHunter.Core.Models
    {
        public class BaseVulnerabilityData
        {
            public string Guid { get; set; }
            public string Description { get; set; }
        }
    }
    namespace NVDHunter.Core.Models
    {
        public class Filter
```

```
    {
        public string startDate { get; set; }
        public string endDate { get; set; }
        public int pageNumber { get; set; } = 1;
        public int resultsPerPage { get; set; } = 20;
    }
}
namespace NVDHunter.Core.Models
{
    public class VulnerabilityAssessment
    {
        public string CVE { get; set; }
        public double? NVDBaseScore { get; set; }
        public double? BaseScore { get; set; }
        public double? TemporaryScore { get; set; }
        public string Assessment { get; set; }
    }
}
```