ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ
Кафедра Інформаційної та кібернетичної безпеки

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти магістр

на тему: «**ДОСЛІДЖЕННЯ ТЕХНОЛОГІЇ ЗАХИСТУ ВІРТУАЛЬНОГО ХМАРНОГО СЕРЕДОВИЩА НА ОСНОВІ ГРАФА АТАК»**

Виконав: студент 6 курсу, групи ІКБМ-61
спеціальності 125 Кібербезпека
Вишнівський В.В.
(прізвище та ініціали)
Керівник: Гайдур Г.І.
(прізвище та ініціали)
Рецензент:
(прізвище та ініціали)
Нормоконтроль:
(прізвище та ініціали)

Київ – 2021

РЕФЕРАТ

Текстова частина магістерської роботи: 65 с., 20 рис., 15 джерела.

Віртуалізація, хмарне середовище, хмарні технології, граф атаки, граф віртуальна машина.

Об`єкт дослідження – процес реконфігурації хмарного середовища.

Предмет дослідження – моделі та методи забезпечення захисту віртуальних хмарних ресурсів.

Мета роботи – підвищення захищеності віртуальних хмарних ресурсів на основі моніторингу їх стану, оцінка напрямків протидії вторгнень.

Методи дослідження – теорія графів, теорія алгоритмів, теорія випадкових процесів.

Проведено аналіз ефективності хмарних технологій. На основі проведеного аналізу показано перспективність їх використання, що надає ряд переваг перед традиційними технологіями. Визначено завдання, яке полягає в тому, щоб створити ефективну систему виявлення і реагування на зовнішні впливи, для своєчасного зведення до мінімуму наслідків порушення захисту хмарних віртуальних ресурсів.

Удосконалено математичні моделі для забезпечення захисту віртуальних хмарних ресурсів для програмно-конфігурованих мереж, а саме: математичну модель впливу атаки на віртуальні хмарні ресурси, математичну модель оцінки стану віртуальних хмарних ресурсів, математичну модель вибору контрміри на основі комплексного показника для програмно-конфігурованих мереж. На основі отриманих математичних моделей розроблено граф атак на віртуальне хмарне середовище. Цей граф дозволяє отримати інформацію про всі відомі вразливості системи.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

ПОДАННЯ
ГОЛОВІ ДЕРЖАВНОЇ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ МАГІСТЕРСЬКОЇ РОБОТИ

Направляється студент Вишнівський В.В. до захисту магістерської роботи
(прізвище та ініціали)
за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)
на тему: «ДОСЛІДЖЕННЯ ТЕХНОЛОГІЇ ЗАХИСТУ ВІРТУАЛЬНОГО ХМАРНОГО СЕРЕДОВИЩА НА ОСНОВІ ГРАФА АТАК»

Магістерська робота і рецензія додаються.

Директор ННІ ЗІ _____ В.А. Савченко
(підпис)

Довідка про успішність

Вишнівський В.В. за період навчання в Навчально-науковому інституті захисту інформації, з 2020 року до 2022 року повністю виконав навчальний план за напрямом підготовки, спеціальністю з таким розподілом оцінок за:
національною шкалою: відмінно ____%, добре ____%, задовільно ____%;
шкалою ECTS: A ____%; B ____%; C ____%; D ____%; E ____%.

Методист ННІ ЗІ_____ Алексіна Л.Т.
(підпис) (прізвище та ініціали)

Висновок керівника магістерської роботи

Студент Вишнівський В.В. під час виконання магістерської роботи показала відмінну теоретичну та практичну підготовку. Проявила самостійність, ініціативність, високу старанність, добрі знання теоретичного матеріалу по спеціальним дисциплінам, креативність при рішенні типових інженерних рішень та вміння правильно використовувати науково-технічну літературу.

Все це дозволяє оцінити виконану магістерську роботу студента Вишнівського В.В. на оцінку «відмінно» та присвоїти кваліфікацію магістр з інформаційної та кібернетичної безпеки.

Керівник роботи _____ Гайдур Г.І.
(підпис)
“ ___ ” _____2021 року

Висновок кафедри про магістерську роботу

Магістерську роботу розглянуто. Студент Вишнівський В.В. допускається до захисту даної роботи в Державній екзаменаційній комісії.

Завідувач кафедри Інформаційної та кібернетичної безпеки

_____ Гайдур Г.І.
(підпис) (прізвище та ініціали)
“_____” _____2021 року

ВІДГУК РЕЦЕНЗЕНТА
на магістерську роботу

студента(ки) Вишнівський Віктор Вікторович
на тему: «ДОСЛІДЖЕННЯ ТЕХНОЛОГІЇ ЗАХИСТУ ВІРТУАЛЬНОГО ХМАРНОГО СЕРЕДОВИЩА НА ОСНОВІ ГРАФА АТАК»

Актуальність:

Магістерська робота студента Вишнівського Віктора Вікторовича присвячена дослідженню технології захисту віртуального хмарного середовища на основі графа атак. Хмарні технології відомі, як нові і потужні обчислювальні парадигми. Це нове покоління мережевої обчислювальної моделі забезпечує програмне і апаратне забезпечення, як на вимогу її ресурсів і різних послуг через Інтернет. Проте, проблема безпеки не дозволяє користувачам приймати хмарні рішення для виконання вимог IT для багатьох критично важливих бізнес-обчислень. Тому тема магістерської роботи є актуальною.

Позитивні сторони:

1. Зміст магістерської роботи відповідає завданню. Робота, яка виконана Вишнівським Віктором Вікторовичем, має високий рівень знань і ступінь підготовленості до майбутньої роботи з фаху.
2. Поставлені в магістерській роботі задачі з огляду, аналізу, дослідження, виконано в повному обсязі.
3.Достатньо успішно викладені технічні питання.
4 Текст викладений грамотно, ясно, послідовно. Графічний матеріал оформлений якісно. Широко використовується науково-технічна література.

Недоліки

1 Зі змісту роботи не зрозуміло, чи можна доповнити систему невідомими атаками.
2 Доцільно було б розглянути потенційні сфери застосування запропонованої технології.

Висновки: Незважаючи на недоліки магістерська робота заслуговує оцінку *відмінно*, а студент Вишнівський Віктор Вікторович присвоєння кваліфікації магістр з Інформаційної та кібернетичної безпеки.

Якість бакалаврської роботи	
виконано на замовлення підприємства	
виконано за тематикою НДР	
виконано з макетом	
має практичну цінність	+

Підпис рецензента

_____
(посада, науковий ступінь, вчене звання)

_____ _____
(підпис) (прізвище та ініціали)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AMD-V (AMD Virtualization) – процесори підтримують апаратну віртуалізацію
ARP (Address spoofing чи Address Resolution Protocol) – мережевий протокол, призначений для перетворення IP адрес (адрес мережевого рівня) в MAC-адреси
CPU (central processing unit) – центральний процесор
CVE (Common Vulnerabilities and Exposures) – ідентифікатори загальних вразливостей
CVSS (Common Vulnerability Scoring System) – загальна система оцінки вразливостей.
DDoS-атаки (Distributed Denied of Service) – розподілена атака на відмову в обслуговуванні
DG (Default Gateway) – шлюз за замовчанням
DPI (Deep Packet Inspection) – технологія накопичення статистичних даних, перевірки і фільтрації мережевих пакетів по їх вмісту.
ERSPAN (Encapsulated Remote Switch port Analyzer) – інкапсулює весь відзеркалений трафік в пакети.
ETH (Ethernet) – технологія пакетної передачі даних для комп'ютерних мереж
Iaas (infrastructure as a service) – інфраструктура як послуга
IDS (Intrusion Detection Systems) - Система виявлення вторгнень
Intel-VT (Intel Virtualization Technology) – Апара́тна віртуаліза́ція
IPS (intrusion prevention system) – система попередження вторгнень
IP адреса (Internet Protocol Address) – унікальна мережева адреса вузла в комп'ютерній мережі
LACP (link aggregation control protocol) – відкритий стандартний протокол агрегації каналів
MAC (Medium Access Control) – керування доступом до середовища
NIST (The National Institute of Standards and Technology) – національний інститут стандартів та технологій
NOX/POX – OpenFlow контролери,які підтримують мову C++ і Python
NVD (National Vulnerability Database) – національна база даних вразливостей
OFS (OpenFlow Switch) – протокол для управління комутаторами, що формують площину передавання даних.
OSVDB (Open Sourced Vulnerability Database) – відкрита база даних вразливостей .
OVS (Open Virtual Switch) – відкритий віртуальний комутатор
PaaS (Platform as a service) – платформа як послуга
QoS (quality of service) – якість обслговування
RSPAN (Remote Switch Port Analyzer) – те ж саме що SPAN, тільки з можливістю віддаленого доступу
SaaS (Software as service) – програ́мне забезпечення як по́слуга
SDN (Self-Defending Network) – програмно-конфігуровані мережі
SLA (Service Level Agreement) – угода про рівень обслуговування

SPAN (local Switched Port Analyzer) – коммутаторы с поддержкой такой технологии позволяют избирательно анализировать сетевой трафик, предназначенный для любого хоста в сети с использованием анализатора протоколов
TCP/IP (Transmission Control Protocol/Internet Protocol) – набір мережевих протоколів передачі даних
VirtualBox (Oracle BM VirtualBox) – программний продукт віртуалізації для операційних систем Microsoft Windows, Linux, FreeBSD, Mac OS X, Solaris / OpenSolaris, ReactOS, DOS
VLAN (Virtual Local Area Network) – віртуальна локальна мережа
VLAN ID (Virtual Local Area Network identifier) - 12-бітний ідентифікатор VLAN
VNS (Virtual Network Services) – віртуальна мережева служба
VSWITCH (OVS) (Open vSwitch) – програмний багаторівневий комутатор з відкритим вихідним текстом
XCP (Xen Cloud Platform) – платформа для розгортання та управління роботою хмарної інфраструктури
Xen – система віртуалізації
ВМ – віртуальна машина
ГКО – граф кореляції оповіщення
ОС – операційна система
ПЗ – програмне забезпечення
СГА – сценарій графу атаки

ВСТУП

Протягом останніх років хмарні обчислення стали новою парадигмою для хостингу і надання послуг через Інтернет. З метою зниження загальної вартості системи, все більше бізнес-компаній почали впроваджувати свої бізнес-додатки в центрах обробки даних (ЦОД) і розмістили там свої інфраструктурні ресурси для бізнес-операцій. Проте, технології хмарних обчислень все ще розвиваються, багато питань необхідно вирішити в області безпеки. Тому актуальними являються наукові дослідженння в IT-індустрії, які повинні забезпечити ефективну архітектуру і рішення. Як правило, моделі послуг, що надаються для хмарних обчислень можуть бути розділені в три категорії [7, 45]: software as service (SaaS), platform as a service (PaaS) та infrastructure as a service (Iaas).

Ці три моделі обслуговування, мають різний рівень вимог до безпеки і стикаються з багатьма різними проблемами безпеки [34]. У моделі SaaS, дані різних користувачів зберігаються в центрі обробки даних провайдерів хмарних послуг в одному фізичному ресурсі. Питання безпеки включають в себе: захист даних, поділ даних, контроль доступу до даних, управління ідентифікацією і процес реєстрації користувача. Сервісна модель PaaS дозволяє користувачам або розробникам створювати свої власні додатки до необхідної платформи. Питання безпеки в моделі PaaS включає в себе програмне забезпечення безпеки, уразливості і управління програмним забезпеченням. У цьому випадку користувачі сервісної моделі IaaS мають кращийх контроль над безпекою на необхідних ресурсах. Постачальник хмарних сервісів тільки опікується проблемами безпеки до гіпервізора. Проте, безпека віртуалізації є найбільш важливою проблемою для багатокористувальницького середовища орендарів. IaaS є основою всіх хмарних сервісів, тому безпека та ризик на цьому рівні істотно впливає на інші моделі обслуговування.

Зловмисники проявляють винахідливість до компроментації віртуальних машин шляхом експлуатації вразливостей програмного забезпечення та управління

віртуальними машинами. [7]. Такі напади є ефективними в хмарному середовищі, так як обчислювальні ресурси розподіляються між користувачами.

Вони, як правило запускають приховане сканування, яке також може бути частиною нормальної діяльності хмари, а потім за допомогою програми копіюють файли віртуальних машин. Наприклад, зловмисники можуть досліджувати уразливості віртуальних машин і використовувати їх для розгортання подальших атак.

Для захисту віртуальних машин в загальнодоступних хмарах, системні адміністратори мають повний контроль над хост-машинами, тому вразливість може бути виявлена і виправлена системним адміністратором в централізованому порядку. Тим не менш, усунення відомих дірок в безпеці, де користувачі хмари мають привілей управляти встановленим програмним забезпеченням на їх керованих віртуальних машинах, не може працювати ефективно.

Описані вище ситуації в області безпеки і сервісу є серйозною проблемою. Потрібен постійний суворий контроль безпеки і заходи захисту, щоб забезпечити захист від внутрішніх і зовнішніх атак.

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ЗАХИСТУ ХМАРНОГО СЕРЕДОВИЩА

1.1 Стан і перспективи розвитку хмарних технологій

Стрімкий розвиток ІТ, зокрема, впровадження хмарних технологій (ХТ) стрімко зростає. Хмарні технології (Cloud Computing) — це парадигма, що передбачає віддалену обробку та зберігання даних. Хмара — сервер або мережа, де зберігаються дані та програми, що з'єднуються з користувачами через Інтернет. ХТ дозволяють споживачам використовувати програми без установки і доступу до особистих файлів з будь-якого комп'ютера, що має доступ в Інтернет.

ХТ відкривають новий підхід до обчислень, при якому обладнання та програмне забезпечення не належать підприємству. Замість цього провайдер надає замовнику вже готовий сервіс [26].

До сучасних хмарних обчислень встановлено наступні вимоги:

- самообслуговування на вимогу;
- універсальний доступ по мережі;
- об'єднання ресурсів;
- еластичність;
- облік споживання.

З точки зору постачальника, завдяки об'єднанню ресурсів та непостійному характеру використання з боку споживачів, ХТ дозволяють економити на масштабах, використовуючи менші апаратні ресурси, ніж при виділенні апаратних потужностей для кожного споживача, а за рахунок автоматизації процедур модифікації виділення ресурсів істотно знижуються витрати на абонентське обслуговування.

З точки зору споживача, ці характеристики дозволяють отримати послуги з високим рівнем доступності і низькими ризиками непрацездатності, забезпечити швидке масштабування обчислювальної системи завдяки еластичності без

необхідності створення, обслуговування і модернізації власної апаратної інфраструктури.

Використання ХТ дає ряд переваг перед традиційними технологіями ІТ:

- організація може більш ефективно управляти використанням обчислювальних ресурсів;
- підвищується керованість ІТ-інфраструктурою;
- спрощується управління безперебійністю роботи організації, завдяки закладеним в концепцію системам резервного копіювання і міграції віртуальних машин;
- скорочення витрат на ІТ-інфраструктуру, таких як парк обчислювальних ресурсів, електроенергію, а також персонально обслуговує цю інфраструктуру.
- не потрібні потужні комп'ютери та сервери;
- менше витрат на закупівлю програмного забезпечення і його систематичне оновлення;
- необмежений обсяг збереження даних;
- доступність з різних пристроїв і відсутня прив’язка до робочого місця;
- забезпечення захисту даних від втрат та виконання багатьох видів навчальної діяльності, контролю і оцінювання, тестування он-лайн, відкритості освітнього середовища;
- економія коштів на утримання технічних фахівців.

Загальною перевагою для всіх користувачів ХТ є те, що отримати доступ до «хмари» можна не лише з ПК, але також з нетбука, смартфона, планшета, тому що головною вимогою для доступу є наявність Інтернету, а для роботи ПЗ «хмари» використовуються потужності віддаленого серверу.

За оцінками експертів, використання ХТ в багатьох випадках дозволяє скоротити витрати в два-три рази в порівнянні з утриманням власної розвиненої ІТ-структури. Також, головною перевагою використання ХТ є можливість швидко пристосовуватись до змін у середовищі будь-якої установи в умовах стрімкого розвитку всіх галузей науки і техніки, є актуальним [72].

Доцільно відмітити і деякі недоліки ХТ. Одним з суттєвих недоліків являється необхідність доступу до швидкісного Інтернету. Стрімке збільшення кількості Інтернет-провайдерів та постійне покращення якості Інтернет-послуг мали б вирішити цю проблему, проте можливі перебої у роботі чи неполадки у провайдерів, можуь призвести до зупинки роботи відділів чи цілих підприємств на незначний час [7]. Також до недоліків ХТ можна віднести обмежену функціональність ПЗ при роботі з ними через Інтернет.

Існують проблеми з боку безпеки, коли хмарні провайдери можуть роками зберігати важливу інформацію на своїх серверах, а кібер-злочинці – перехоплювати інформацію. Звісно, великі хмарні провайдери застосовують всі можливі засоби для забезпечення максимальної безпеки інформації і вкладають кошти в розробку нових ефективних засобів захисту, проте поки що не варто зберігати чи передавати особливо важливі документи в «хмари».

За оцінками світових компаній, хмарні технології визначають розвиток IT-індустрії в найближчі 5-8 років і 80% зростання світового ринку IT буде припадати саме на частку «хмар».

Плани використання хмарних рішень українськими підприємствами, а також інтенсивне освоєння технології IT-компаніями створюють потенціал ринку, який до 2025р. забезпечить експоненціальне зростання, характерне для хмарних ринків розвинених країн.

Ринок ХТ стрімко росте. На ньому активно пропонуються послуги як для фізичних осіб, так і корпорацій. За прогнозами компанії IDC середньорічний темп приросту світового ринку хмарних сервісів з 2013 року по 2021 рік склав 26,4%, що в п’ять разів перевищує темпи росту IT-індустрії в цілому [33].

На сьогодні 70% організацій в світі вже або використовують хмарні технології, або досліджують це питання. Згідно з дослідженням, 25% респондентів готові йти в «хмари» заради зниження IT-витрат.

Зараз спостерігається поступова міграція освітніх сервісів за допомогою сучасних інформаційно-комунікаційних технологій та інформаційних ресурсів в

хмару, що згодом приведе до стрімкого впровадження цих сервісів в освіту і соціальну сферу.

Упровадження ХТ є новим напрямом у сфері комп’ютерних технологій, що розвивається, і потребує подальшого дослідження.

1.2 Аналіз сучасних технологій віртуалізації

Віртуалізація є основною технологією для сервісної моделі IaaS та для створення багатокористувацького середовища. Постачальник послуг створює виділений віртуальний ресурс інфраструктури для кожного орендаря. Ці віртуальні обчислювальні мережеві середовища називаються віртуальними машинами (ВМ) та віртуальними мережами (VNs). Користувачі можуть запускати свої власні додатки, операційні системи або системні служби в цих логічних, різних середовищах, без прив'язки до будь-якої конкретної фізичної комп'ютерної системи. Оператор середовища, що віртуалізує, сфокусований на створенні додатків, служб та примірників, які працюють на різних фізичних комп'ютерних системах [12]. Розглянемо основні види технології віртуалізації.

Віртуалізація на основі Гіпервізора. Гіпервізор, також відомий як монітор віртуальних машин (ВММ), являє собою невелику частину програмного забезпечення або програмно-апаратного, який працює поверх апаратного забезпечення машини. Основними функціями гіпервізора є:

- визначення і реагування на захищені або привілейовані CPU операції, що проведені кожною віртуальною машиною;
- адміністрування черг, диспетчеризація, повернення результатів апаратних запитів від віртуальних машин.

Адміністративна операційна система, відома також як привілегійований домен, працює поверх гіпервізора, так само, як віртуальні машини. Ця адміністративна операційна система відповідає за управління віртуальними машинами на одному сервері і працює з гіпервізором [12].

Розрізняють два типи гіпервізора. Гіпервізор I типу просто потрібно запустити над апаратними засобами для управління апаратними засобами і управління гостьовою операційною системою. Він також відповідає за більшість комунікацій між усіма гостьовими ОС і апаратними засобами. Відомий екземпляр цього типу Xen, BMware ESXi і Microsoft Hyper-V. Гіпервізор типу II працює як додаток в рамках хоста операційної системи. ОС хоста відповідає за надання драйверів введення / виведення і управління гостьовою ОС віртуальних машин.

BMware Workstation, BMware Server і Virtual Box є прикладом архітектури віртуалізації типу II на основі гіпервізора.

Повна віртуалізація. Повна віртуалізація використовує гіпервізор, але включає в себе код в гіпервізорі, який емулює базове обладнання при необхідності, дозволяючи не модифікованим операційним системам працювати поверх гіпервізора [12]. Одним з відомих програмних забезпечень повної віртуалізації є BMWare ESXi, який використовує версію Linux (відомий як Console Service) в якості своєї адміністративної операційної системи. Гіпервізор на сервері BMWare ESXi називається BMkernel, що представляє собою гіпервізор товстого шару, який включає драйвер апаратного забезпечення BMMs для кожної віртуальної машини.

З повною віртуалізацією, не модифікована ОС проводить програму користувача, що емулює машину, на якій працює гостьовий ОС. Він також має перевагу в тому, що віртуалізована архітектура може повністю відрізнятися від приймаючої архітектури. Наприклад, QEMU може імітувати процесор MIPS на хості IA-32 і багато інших чипів [13]. Проте, цей рівень апаратної незалежності відбувається за рахунок штрафу в швидкості.

Паравіртуалізація. Цей тип віртуалізації Xen. Xen забезпечує дуже маленький і компактний гіпервізорний шар, який працює безпосередньо поверх апаратного забезпечення, а також надає послуги віртуалізованій ОС. Цей тонкий шар гіпервізора відрізняється від товстого шару повною віртуалізацією. З Xen тільки гіпервізор володіє повними правами і він покладається на надійність оцінки ОС, щоб забезпечити апаратні драйвери, ядро і призначений для користувача простір.

Цей домен управління називається «доменом 0» (dom0), який використовує власне ядро Linux для підтримки свого адміністративного середовища.

«Домен 0» є першим доменом створеним автоматично при завантаженні системи і він має особливі привілегії управління, делеговані гіпервізору. Цей привілегійований домен дозволяє гіпервізору отримувати доступ до пристроїв і виконувати функції управління. Всі інші гостьові домени (domUs) є непривілегійовані домени і управляються Dom0. Апаратне середовище для всіх гостей не змодельоване, воно виконуються в їх власних ізольованих доменах, ніби вони працюють на окремій системі. Проте, гостьова ОС повинна бути спеціально модифікована, щоб працювати в цьому середовищі. Паравіртуалізація може забезпечити підвищення продуктивності в порівнянні з іншими підходами, оскільки модифікації в ОС дозволяють операційній системі безпосередньо зв'язуватися з гіпервізором і, отже, не несуть ніяких накладних витрат, пов'язаних з емуляцією, які присутні для інших віртуалізацій на базі гіпервізора.

Віртуалізація на рівні ядра. Для цього типу віртуалізації не потрібен гіпервізор, але він працює на спеціально модифікованому ядрі Linux, яке містить розширення, призначене для управління декількома віртуальними машинами, кожна з яких містить гостьову операційну систему.

Приклади технологій віртуалізації на рівні ядра включають в себе користувальницький режим Linux (UML) і ВМ на основі ядра (KBM). UML підтримувався в ядрах Linux досить довгий час, але вимагав спеціальної збірки ядра Linux для гостьових операційних систем і тому була введена KBM 2.6.20. UML не потребує будь-якого окремого адміністративного програмного забезпечення для виконання або управління віртуальними машинами, які можуть бути виконані з командного рядка Linux. KBM використовує драйвер пристрою в ядрі хоста для зв'язку між основним Linux ядром і віртуальною машиною. Вимагає підтримки процесорів для віртуалізації (Intel VT або AMD-V), і використовує модифікований процес QEMU як дисплей виконання контейнера для своїх віртуальних машин. Багато в чому віртуалізації KBM на рівні ядра є спеціалізованою версією повної віртуалізації, де ядро Linux служить в якості гіпервізора [12].

Віртуалізація з апаратною підтримкою. Віртуалізація з апаратною підтримкою – це спосіб підвищення ефективності апаратної віртуалізації. Вона включає в себе використання спеціально розроблених процесорів і апаратних компонентів, які допомагають поліпшити експлуатаційні характеристики в гостьовому середовищі [12]. Гіпервізор на основі систем, такі як Xen і BMWare ESXi, а також технології віртуалізації на рівні ядра, такі як KBM, можуть скористатися апаратною підтримкою віртуалізації. Останнє покоління Intel (Intel-VT) і AMD (AMD-V) процесори підтримують апаратну віртуалізацію. Віртуальні машини в середовищі віртуалізації можуть працювати на не модифікованих операційних системах, оскільки гіпервізор може використовувати підтримку обладнання для віртуалізації та обробки привілейованих і захищених операцій і запитів доступу до обладнання, а також спілкуватися і керувати ВМ.

1.3 Проблеми компроментації інформації в віртуальному середовищі

У хмарній системі уразливості для каналу зв'язку можуть надходити з наступних компонентів: віртуальних машин, сервера хмари і від підключень до фізичної мережі. Нижче наведені можливі уразливості.

Безпека канального рівня можна вважати синонімом безпеки локальної мережі. Як правило, атаки на канальному рівні припускають, що атакуючий знаходиться в локальній мережі, або є якийсь посередник, який навмисно або ненавмисно допомагає виконанню атак.

Завдання атакуючого отримати доступ до певних ресурсів, інформації або, як мінімум, порушити нормальну роботу мережі.

Небезпека проблем в безпеці на канальному рівні полягає в тому, що зламавши мережу на канальному рівні, атакуючий може переступити через засоби захисту на більш високих рівнях.

Як правило, атаки виконуються в комплексі, а не по одній. Вдале виконання атаки може бути не метою, а засобом отримати чогось більшого.

Наприклад, атака ARP-spoofing дозволяє атакуючому перехоплювати весь трафік між ресурсом і якимось користувачем (або підміняти його). Однак, виконати цю атаку можна тільки в межах широкомовного сегмента мережі. Якщо в мережі присутній комутатор Cisco з настройками за замовчуванням, то можна підняти тегованих канал між комп'ютером атакуючого і комутатором і отримати таким чином доступ до інших широкомовною сегментам мережі. В такому випадку виконати ARP-spoofing можна в усіх доступних місцях сегментів.

Аналіз атак на канальному рівні мережі. Залежно від результату, який буде отриманий, атаки можна розділити на такі кілька типів (виділені типи атак не охоплюють всі можливі атаки):

Людина посередині (Man in the middle, MitM) - атакуючий знаходиться між двома жертвами і або прослуховує трафік, який передається між ними, або перехоплює його і підміняє;

Відмова в обслуговуванні (DoS) - атака на якийсь ресурс системи з метою довести його до відмови. На канальному рівні є ряд атак, які дозволяють отримати односторонній доступ до якогось ділянці мережі;

Несанкціонований доступ до мережі - використовуючи якісь недоліки в протоколах, отримання доступу до тих ділянок мережі, які теоретично повинні були бути недоступні;

Порушення роботи мережі або її ділянок - деякі недоліки протоколів канального рівня не можуть використовуватися передбачуваним чином.

Не всі протоколи, які будуть розглядатися, відносяться до канального рівня. Однак, саме особливості роботи цих протоколів канального рівня, дають можливість маніпулювати цими протоколами. Тому часто відповідні атаки і засоби захисту відносять до канального рівня мережі.

Поширені атаки канального рівня:

- ARP-spoofing (ARP-poisoning) - техніка мережевої атаки, що застосовується переважно в Ethernet, але можлива і в інших, що використовують протокол ARP мережах, заснована на використанні недоліків протоколу ARP і дозволяє

перехоплювати трафік між вузлами, які розташовані в межах одного широкомовного домену;

- MAC-spoofing - атака канального рівня, яка полягає в тому, що на мережевої карти змінюється MAC-адресу, що змушує комутатор відправляти на порт, до якого підключений зловмисник, пакети, які до цього він бачити не міг;
- Переповнення таблиці комутації - атака заснована на тому, що таблиця комутації в комутаторах має обмежений розмір. Після заповнення таблиці, комутатор не може більше вивчати нові MAC-адреси і починає працювати як хаб, відправляючи трафік на всі порти;
- Атаки на DHCP - це може бути підміна DHCP-сервера в мережі (тоді атакуючий може призначати додаткові параметри DHCP, такі як шлюз за замовчуванням) або атака DHCP starvation, яка змушує DHCP-сервер видати всі існуючі на сервері адреси зловмисникові;
- VLAN hopping - несанкціоноване отримання доступу до VLAN;
- Атаки на STP - відправлення повідомлень BPDU для зміни поточної топології STP.

Існує безліч різних утиліт, які дозволяють виконувати атаки на мережу. Деякі дозволяють моделювати тільки певні атаки, інші - генерувати будь-який пакет.

Розглядаються такі аспекти використання мережевих утиліт:

- тестування функцій і засобів безпеки;
- тестування вразливостей деяких налаштувань мережевих пристроїв і хостів.

Як правило, коли функція безпеки застосовується одна, то рідко виникають якісь проблеми. Досить просто налаштувати її і перевірити коректність роботи. Набагато складніше налаштувати узгоджену роботу декількох засобів захисту, залишивши при цьому мережа працездатною.

Після настройки функцій безпеки обов'язково необхідно перевірити коректність її роботи. По-перше, тому що бувають випадки, коли функція працює не зовсім так, як було заявлено виробником обладнання або не зовсім так як хотілося б. По-друге, тому що функції впливають один на одного і додавання нової функції може привести до змін в поведінці іншої функції.

Крім тестування функцій безпеки утиліти можуть використовуватися для перевірки вразливості деяких налаштувань мережевих пристроїв і хостів.

Наприклад, на багатьох комутаторах використовуються такі параметри, які дозволяють виконати ряд атак і отримати несанкціонований доступ до мережі або вивести з ладу мережеві пристрої. Використання утиліт в даному випадку демонструє простоту злому мережі і вразливість таких налаштувань комутаторів.

Yersinia - мережева утиліта, яка дозволяє експлуатувати недоліки ряду протоколів, що використовуються активним мережевим обладнанням і стандартних мережевих протоколів, що використовуються в мережі. На даний момент можливі атаки на:

стандартні мережеві протоколи: Dynamic Host Configuration Protocol (DHCP), Spanning Tree Protocol (STP), 802.1Q, 802.1X;

пропрієтарні протоколи Cisco: Inter-Switch Link Protocol (ISL), VLAN Trunking Protocol (VTP), Cisco Discovery Protocol (CDP), Dynamic Trunking Protocol (DTP), Hot Standby Router Protocol (HSRP).

У сучасних локальних мережах обмін інформацією, як правило, передбачає передачу даних через комутатор.

Тому сам комутатор і протоколи, які використовують комутатори можуть бути метою атак. Більш того, деякі настройки комутаторів дозволяють виконати ряд атак і отримати несанкціонований доступ до мережі або вивести з ладу мережеві пристрої. Однак, комутатор може бути і досить потужним засобом захисту. Так як через нього відбувається все взаємодія в мережі, то логічно контролювати це на ньому. Звичайно використання комутатора як засобу захисту передбачає, що використовується не найпростіший комутатор 2го рівня, а комутатор з відповідними функціями для забезпечення безпеки.

Port security - функція комутатора, що дозволяє вказати MAC-адреси хостів, яким дозволено передавати дані через порт. Після цього ПОРТ не передає пакети, якщо MAC-адресу відправника не вказано як дозволений. Крім того, можна вказувати не конкретні MAC-адреси, дозволені на порту комутатора, а обмежити кількість MAC-адрес, яким дозволено передавати трафік через порт.

Використовується для запобігання: несанкціонованої зміни MAC-адреси мережевого пристрою або підключення до мережі, атак спрямованих на переповнення таблиці комутації.

Таблиця 1.1

Функції комутаторів для забезпечення безпеки роботи мережі на канальному рівні

Функції коммутаторів	**Захист від атак**
Port security	Переповнення таблиці комутації, несанкціонована зміна MAC-адреси
DHCP Snooping	Підміна DHCP-сервера в мережі, DHCP starvation
Dynamic ARP Inspection	ARP-spoofing
IP Source Guard	IP-spoofing

DHCP snooping - функція комутатора, призначена для захисту від атак з використанням протоколу DHCP. Наприклад, атаки з підміною DHCP-сервера в мережі або атаки DHCP starvation, яка змушує DHCP-сервер видати всі існуючі на сервері адреси зловмисникові.

DHCP snooping регулює тільки повідомлення DHCP і не може вплинути безпосередньо на трафік користувачів або інші протоколи. Деякі функції комутаторів, що не мають безпосереднього відношення до DHCP, можуть виконувати перевірки на підставі таблиці прив'язок DHCP snooping (DHCP snooping binding database). В тому числі: Dynamic ARP Protection (Inspection) - перевірка ARP-пакетів, спрямована на боротьбу з ARP-spoofing, IP Source Guard виконує перевірку IP-адреси відправника в IP-пакетах, призначена для боротьби з IP-spoofingом.

Dynamic ARP Inspection (Protection) - функція комутатора, призначена для захисту від атак з використанням протоколу ARP. Наприклад, атаки ARP-spoofing, що дозволяє перехоплювати трафік між вузлами, які розташовані в межах одного широкомовного домену.

Dynamic ARP Inspection (Protection) регулює тільки повідомлення протоколу ARP і не може вплинути безпосередньо на трафік користувачів або інші протоколи.

IP Source Guard (Dynamic IP Lockdown) - функція комутатора, яка обмежує IP-трафік на інтерфейсах 2го рівня, фільтруючи трафік на підставі таблиці прив'язок DHCP snooping і статичних відповідностей. Функція використовується для боротьби з IP-spoofingom.

Аналіз уразливостей технологій віртуалізації. Віртуалізація є ключовою технологією в хмарних обчисленях. Однією з найбільших проблем, питань безпеки хмарної системи є з'єднання між віртуальними машинами [22]. Можливі уразливості від віртуальних машин включають:

- hopping: процес переходу з однієї віртуальної машини на іншу віртуальну машину;
- escape: зловмисник може получити доступ до через гіпервізор і нападати на іншу частину віртуальних машин;
- mobility: вміст ВМ зберігається в файлі на гіпервізорі. Цей файл можна перемістити або скопіювати в інше місце;
- template: ВМ клонуються за допомогою шаблону для прискорення створення хмарної системи. Якщо при роботі віртуального хмарного середовища використовуються шаблони, зловмисник може змінити налаштування віртуальної машини, щоб компроментувати вс нові віртуальні машини.

Аналіз уразливостей віртуальних мереж. Основні режими роботи паравіртуальної мережі: режим моста і режим маршрутизації dom0 повинні можуть бути скомпроментовані. Слід відмітити, що можливість компрометації dom0 більш важлива, так як ця мережа відповідає за роботу віртуального хмарного середовища в цілому. Тому це ще одне суттєве питання безпеки в мережі віртуального хмарного середовища. Можливі уразливості віртуальної мережі dom0, включають:

- sniffing: у режимі мережевого моста, віртуальна машина здатна проводити моніторинг всього трафіку віртуальної мережі на тому ж мережевому мості, використовуючи Sniffing інструменти, такі як Wireshark;

- spoofing: при цьому підході скомпроментована ВМ може використати протокол дозволу адрес (ARP), який може перехоплювати пакети ВМ, та прослуховувати весь трафік потерпілого ВМ;

- compromised dom0: dom0 може контролювати всі зв'язки між віртуальними машинами і підключатися до зовнішньої мережі. Як тільки dom0 скомпрометували, зловмисник може змінити трафік у віртуальній мережі і заволодіти інформацією.

Аналіз можливостей компроментації локальних комп'ютерних мереж. Ethernet являє собою систему мовлення. Для підвищення безпеки, VLAN реалізована для посилення ізоляції мережі, а також розширення можливості управління системою. Однак з реалізацією VLAN кожне повідомлення як і раніше може охопити всі частини однієї і тієї ж VLAN. Це означає, що повідомлення можуть бути прочитані будь-яким хостом на тій же VLAN. Це дозволяє зловмисникові підслуховувати пакети, що проходять через мережу. Ethernet не передбачає механізму для перевірки справжності відправника. Це дозволяє зловмисникові генерувати пакети або відтворювати раніше підслухані [23].

Відомі атаки проти кінцевих хостів в мережі рівня 2 засновані на Medium Access Control (MAC), Address spoofing чи Address Resolution Protocol (ARP).Це формує основу багатьох атак, таких як DoS і Man-In-The-Middle. Інші можливі атаки в VLAN на основі мережі [24] включають:

- MAC flooding attack: це напад пов'язаний з обмеженням працюючих комутаторів і мостів. Вони володіють кінцевою таблицею апаратних засобів для зберігання вихідних адрес всіх прийнятих пакетів. Коли ця таблиця заповнюється, трафік, який спрямований на адреси, які не можуть бути вилучені, буде постійно утилізувати мережу на максимумі її пропускної здатності;

- 802.1Q tagging attack: ця атака відбувається через неправильні настройки комутаторів, які встановлюють порт комутатора в якості небажаного порту. Це зазвичай називають "VLAN leaking", що дозволяє користувачеві VLAN, дістати несанкціонований доступ до іншої VLAN;

- Double-Encapsulated 802.1Q VLAN атаки: ці атаки відбуваються через неправильну настройку комутаторів, які встановлюють рідний ідентифікатор

VLAN ID магістрального порту так само як VLAN ID порту доступу зловмисника. Зловмисник може інкапсулювати ідентифікатор VLAN ID в якості внутрішнього шару протоколу 802.1Q. Після того, як зовнішній тег відігнаний та неправильно обрано параметр комутатора, комутатор просто пересилає пакети за призначенням підробленого внутрішнього тега. Таким чином, зловмисник може отримати несанкціонований доступ до іншої VLAN;

- ARP атаки: ці атаки пов'язані з ARP-запитом і відповіддю, які несуть інформацію про MAC-адресу і ідентифікатор IP адресу хоста. Зловмисник може обдурити комутатор в перенаправленні пакетів на пристрій або хост в іншій мережі VLAN, посилаючи ARP-пакети, що містять підроблену інформацію.

1.4 Висновки до розділу 1

Таким чином, проведені дослідження показали, що найбільш важливим фактором для користувачів хмари є безпека. Дослідження Cloud Alliance Security (CSA) показують, що серед усіх проблем несанкціонований доступ і використання хмарних обчислень розглядається як загроза безпеці, в яких зловмисник може використовувати уразливості в хмарі і її системи ресурсів для розгортання атак. У традиційних центрах обробки даних, де системні адміністратори мають повний контроль над хост-машинами, вразливість може бути виявлена і виправлена системним адміністратором в централізованому порядку. Проте, латання відомих дір безпеки в центрах обробки даних хмари, де користувачі хмари, як правило, мають привілей управляти встановленим програмним забезпеченням на їх керованих віртуальних машинах, не може працювати ефективно. Крім того, користувачі можуть встановити на хмару вразливе програмне забезпечення на їх віртуальних машинах, що істотно сприяє проломам в безпеці хмари. Завдання полягає в тому, щоб створити ефективну систему виявлення атак і реагування, для точного виявлення атак і зведення до мінімуму наслідків порушення надійності та безпеки для користувачів хмари.

Необхідно розробити методику забезпечення безпеки віртуальних хмарних ресурсів на базі виявлення атак та реконфігурації віртуальних мереж в віртуальному хмарному середовищі.

Тому забезпечення безпеки віртуального хмарного середовища включає в себе декілька основних етапів:

- розробку математичної моделі впливу атаки на віртуальні хмарні ресурси, на основі якої розроблено алгоритм аналізу загроз у хмарному середовищі, що дасть змогу отримати маршрут проходження атаки, за допомогою графа атаки та графа кореляції оповіщення, на віртуальне хмарне середовище;
- розробку математичної моделі оцінки стану віртуальних хмарних ресурсів, на основі якої буде розроблено алгоритм вибору контрміри, в результаті роботи якого розраховується показник захисту віртуальних ресурсів та обираються відповідні контрзаходи для підвищення показника захисту віртуальних хмарних ресурсів.
- побудову алгоритмів: аналізу загроз у хмарному середовищі, результатом роботи якого є отримання одного або декількох шляхів проходження атаки на віртуальне середовище; вибору контрміри, який визначає вибір контрзаходів для конкретного сценарію атаки.

2. РОЗРОБКА МАТЕМАТИЧНИХ МОДЕЛЕЙ ЗАХИСТУ ВІРТУАЛЬНИХ ХМАРНИХ РЕСУРСІВ ДЛЯ ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖ

Даний розділ присвячено розробці математичних моделей для забезпечення захисту віртуальних хмарних ресурсів.

Модель впливу атаки на віртуальні хмарні ресурси розроблена для випадків коли зловмисник може знаходитись зовні або всередині системи віртуальної мережі. Головною метою зловмисника являється використання вразливостей віртуальних машин та їх компроментація.

2.1 Аналіз моделей графа атак

Граф атак – це інструмент моделювання, для ілюстрування всіх можливих шляхів атак, виявлення їх і прийняття відповідних контрзаходів [40]. У графі атак, кожен вузол являє собою вразливості віртуального середовища, або наслідок експлуатації цих вразливостей. Також необхідно розуміти, що нормальна дія протоколу може бути використана, як вразливість. Граф атак допомагає виявиявити потенційну загрозу, а також можливі атаки і відомі вразливості в хмарних віртуальних середовищах.

Протидія спробам порушення захисту віртуальних хмарних ресурсів являється актуальним. Дії зловмисників можуть бути описані графом атак.
Неформально, граф атак – це граф, що представляє різні послідовності дій порушника для досягнення його мети. Такі послідовності дій називаються трасами (шляхами) атак. На рис. 2.1 – 2.3 зображені приклади графів атак. Можна виділити наступні види графів атак:

- граф перерахунку стану (state enumeration graph) (рис. 2.1) – в таких графах вершини відповідають трьом показникам (s, d, a), де s – джерело атаки, d – мета атаки, a – елементарна атака; дуги позначають переходи з одного стану в інший;

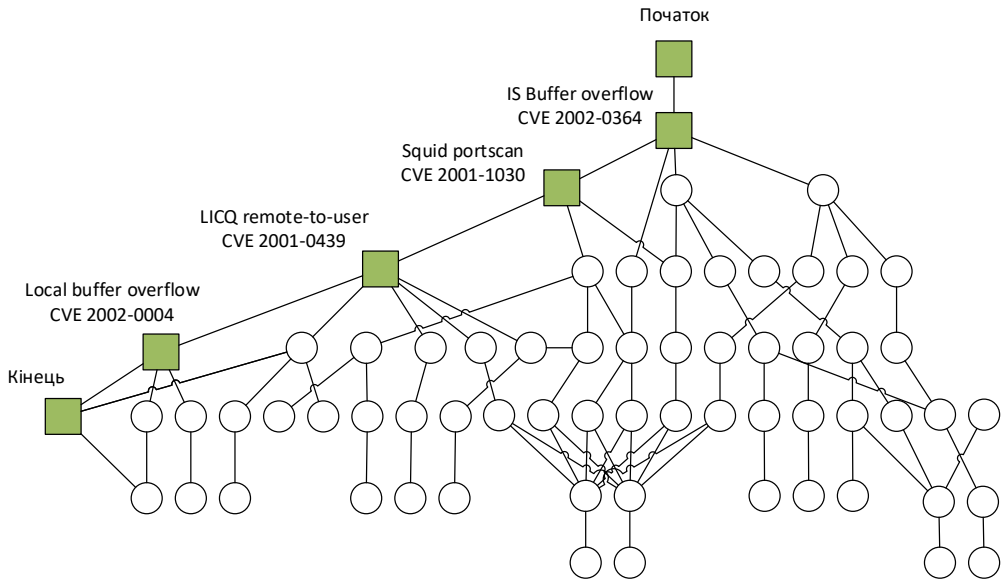

Рис. 2.1 - Граф перерахунку стану

- орієнтований граф залежностей (condition-orienteddependency graph) (рис. 2.2) – вершинам графа відповідають результати атак, а дугам – елементарні атаки, що призводять до таких результатів;

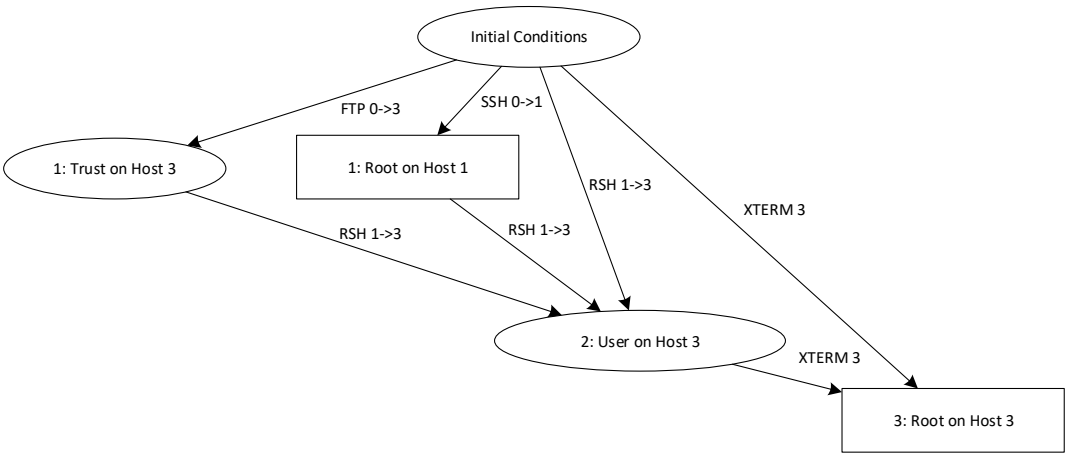

Рис. 2.2 - Орієнтований граф залежностей

- граф залежності використання вразливостей (exploit dependency graph) (рис. 2.3) – вершини відповідають результатом атак або елементарним атакам, дуги відображають залежності між вершинами – умови, необхідні для виконання атаки і наслідок атаки. Наприклад, атака RSH можлива, якщо порушник має

повноваження суперкористувача на хості 1 і хост 3 довіряє хосту 1. В результаті атаки порушник отримує привілеї користувача на хості 3.

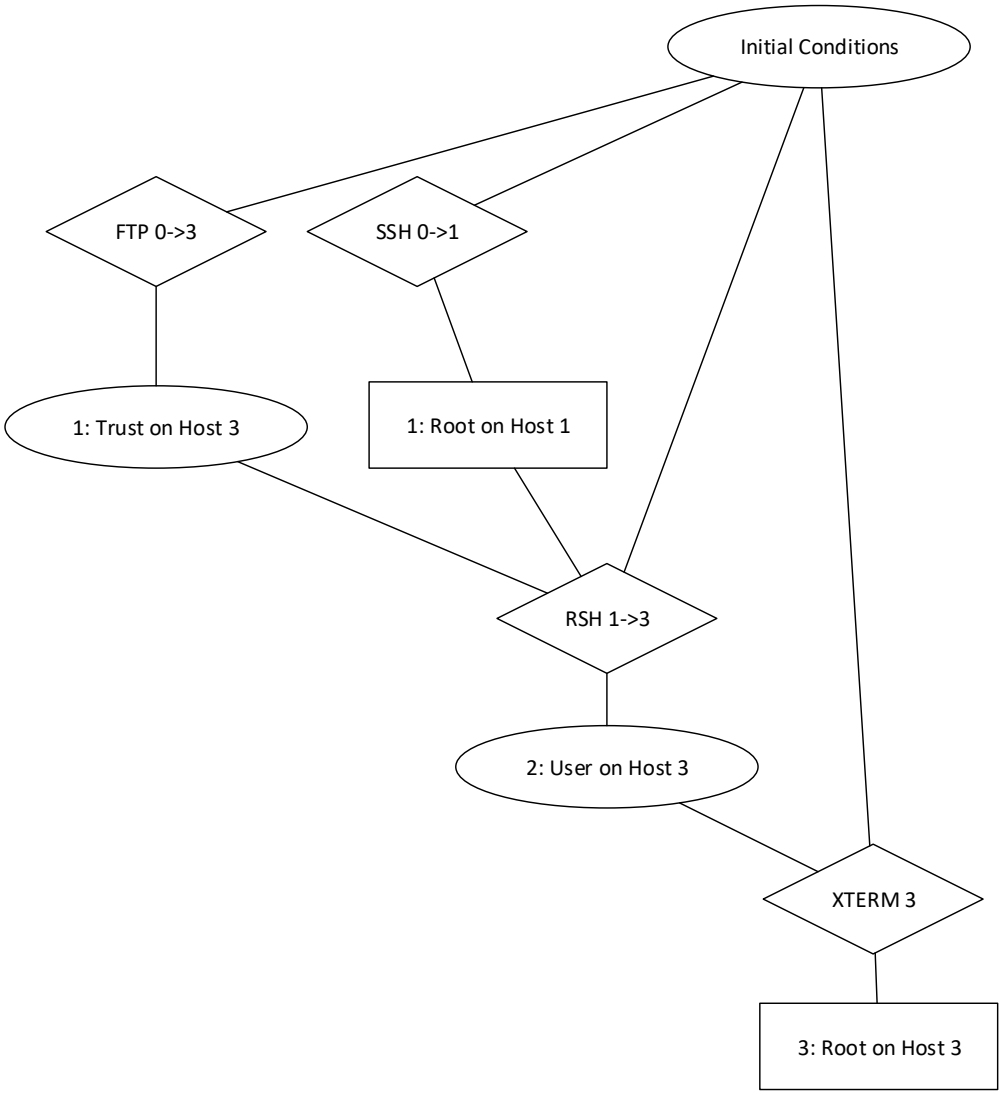

Рис. 2.3 - Граф залежності використання вразливостей

Під елементарної атакою (atomic attack) розуміють використання порушником уразливості.

При синтезі графа атак виникають наступні завдання: формалізація поняття атаки, розробка формальної мови моделювання атак і комп'ютерної системи (що включає порушника, його цілі, мережі, засобів захисту, ставлення досяжності хостів і т.д.), вибір або розробка засобів побудови графа атаки і його візуалізації, розробка засобів автоматизації побудови та аналізу графа.

В основному графи атак розглядаються в контексті аналізу захищеності мереж. Зазвичай такий аналіз зводиться до послідовного сканування всіх хостів мережі на наявність відомих вразливостей. Результатом є звіт, який містить перелік знайдених вразливостей і рекомендації по їх усуненню. В даний час поступово впроваджується інша парадигма аналізу захищеності, що враховує "топологію" комп'ютерної системи – взаємозв'язок об'єктів комп'ютерної системи, їх властивостей і характеристик. Такий аналіз захищеності називається топологічним.

Топологічний аналіз захищеності передбачає побудову графа атак на основі результатів сканування мережі, моделі порушника і даних про конфігурацію мережі (фільтрації МЕ, маршрутизації, виявлення атак, досяжності хостів і т.д.) і його аналіз (імовірнісний, мінімізаційні і т.д.).

Побудований на основі аналізу захищеності граф містить всі відомі сценарії атак для досягнення порушником загроз. Результатом його аналізу може бути: перелік успішних атак, які не виявляються IDS; співвідношення реалізованих заходів безпеки і рівня захищеності мережі; перелік найбільш критичних вразливостей; перелік заходів, що дозволяють запобігти використанню вразливостей в ПЗ, для якого відсутні оновлення; найменше безліч заходів, реалізація яких зробить мережу захищеною.

Графи атак також використовуються при розслідуванні комп'ютерних інцидентів, для аналізу ризиків і кореляцій попереджень систем виявлення атак.

Спочатку процес побудови графа атак був ручним, потім були запропоновані різні підходи до автоматизації даного процесу. Ключовою проблемою побудови графа атак є масштабованість – можливість побудови графа атаки для мережі з великим числом хостів і вразливостей. У даній роботі досліджуються можливі підходи до побудови та аналізу графів атак і проблеми, що виникають при цьому.

2.2 Загальний підхід до побудови графів атак

Побудова графа атак являється складним процесом. Тому необхідно враховувати наступні особливості.

Побудова адекватної моделі. Ефективне моделювання включає автоматизацію процесу побудови моделі мережі. При цьому необхідні дані про наявні вразливості, політику маршрутизації, впровадженої політики безпеки і т.д. Зазвичай у всіх топологічних сканерах безпеки присутні підсистеми, взаємодіючі зі сканерами безпеки, МЕ, СОА, маршрутизаторами і іншими засобами для побудови адекватної моделі мережі.

Крім цього, необхідно визначити досяжність між усіма хостами, враховуючи, наприклад, МЕ і маршрутизатори. Найпростіший підхід обчислення доступності вимагає додаткових N^2 кроків перед побудовою графа і полягає в побудові матриці досяжності A розміру $N \times N$, де $a\,[i][j] = 1$ тоді і тільки тоді, коли хост j доступний хосту i. Визначення досяжності пристроїв у великій мережі є важким завданням. Не завжди можливо визначити досяжність пристрою, використовуючи тільки сканери безпеки. Точне визначення досяжності між хостами вимагає аналізу конфігураційних правил МЕ, маршрутизаторів, комутаторів, VPN-шлюзів, персональних МЕ і інших пристроїв. На рис. 2.4 приведена архітектура топологічного сканера безпеки NetSPA.

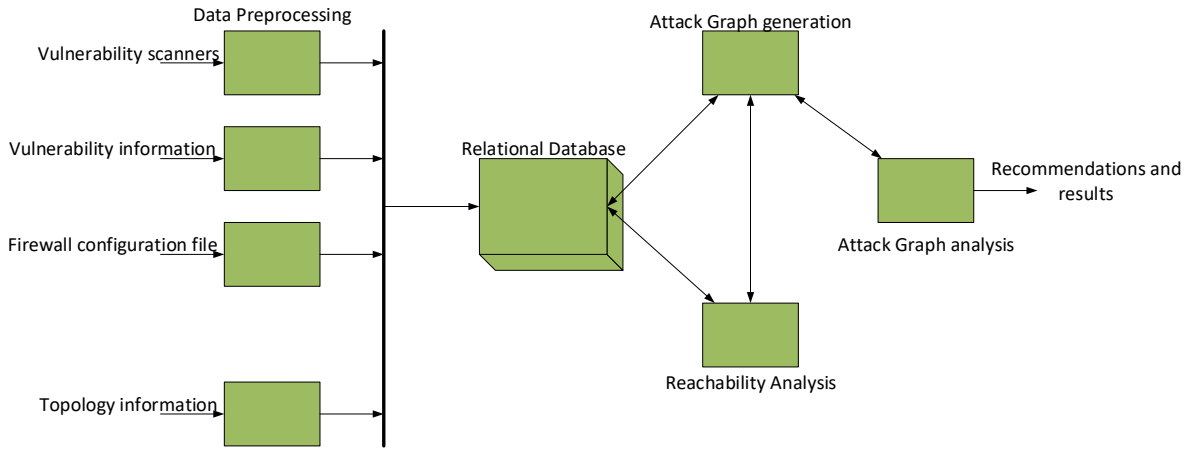

Рис. 2.4 - Архітектура топологічного сканера безпеки NetSPA

Застосування алгоритмів для реальних мереж. Багато алгоритмів побудови графів не можуть бути застосовані для реальних мереж, тому, що всі підходи, які грунтуються на використанні повних графів, є неефективними. Наприклад, повний

граф атак часто не міг бути обчислений в навіть при наявності 13 вразливостей в файлової системі UNIX-хоста [24].

Для підвищення ефективності рішень може використовуватися агрегація подібних хостів, як для поліпшення наочності графа атак, так і для підвищення продуктивності. Найпростіша агрегація полягає в заміні групи ідентичних хостів на один хост. Інший метод полягає в побудові обмеженого графа, що може бути використаний виключно для відповіді на наступні питання: які хости можуть бути скомпрометовані порушником з деякого хоста і яку мінімальну кількість експлойтів дозволить порушнику досягти його мети? Також невідомим параметрами системи можна привласнювати деякі значення за замовчуванням.

В результаті побудови графу атак і його аналізу необхідно видати рекомендації щодо запобігання можливих атак. Дані рекомендації можуть припускати як зміни в мережевій архітектурі, так і установку оновлень ПЗ.

2.3 Побудова графа кореляції оповіщення

Оскільки граф атак надає інформацію про всі відомі вразливості в системі та про підключення, ми отримуємо повну картину поточної ситуації в області захисту системи, де ми можемо спрогнозувати можливі загрози і атаки шляхом кориляції виявлених подій. Якщо подія розпізнається в якості потенційного нападу, ми можемо застосувати конкретні заходи для пом'якшення наслідків впливу нападу або вживати заходи, для запобігання цієї події. Для того, щоб описати атаку і результат її дій, необхідно розробити Сценарій графа атак (СГА).

СГА – є кортежем $G_{СГА} = (V, E)$, де V – множина вершин графа атак, та E – множина спрямованих ребер, що з’єднують вершини графа атак.

Вершини графа атак можуть бути трьох типів: $B_к$ – вузли кон’юнкції (для відображення вразливості), $B_д$ – вузоли диз'юнкції (для позначення результату використання вразливості) і кореневий вузол $B_{кор}$ (для позначення початкової стадії сценарію атаки).

Множина вершин графа атак V визначається наступним чином:

$$V = B_к \cup B_д \cup B_{кор}\ , \qquad (2.1)$$

Множина ребер $e \in E_{pre} \subseteq B_д \times B_к$ відображає те, що $B_д$ повинні бути виконані, щоб досягти $B_к$. Ребро $e \in E_{post} \subseteq B_к \times B_д$ означає, що $B_д$ мають бути отримані, щоб $B_к$ було виконано:

$$E = E_{pre} \cup E_{post}\ , \qquad (2.2)$$

де E_{pre} – ребра, які відображають взаємозв'язок між результатом використання вразливості в попередньому вузлі з самою вразливістю у наступному вузлі, E_{post} – ребра, які відображають взаємозв'язок між вразливостями в попередньому вузлі з можливими результатами використання вразливостей в наступному вузлі.

Граф кореляції оповіщення (ГКО) представляє собою набір вершин таких, що відображають вразливості та тих, що відображають використання цих вразливостей; а також набір орієнтованих ребер, які звязують між собою вершини які є можливими шляхами проходження атаки з врахуванням часових міток, відповідності отриманих оповіщень від мережевого агенту до відповідних вузлів в системі; та відповідності класу цих опоіщень до можливих вразливостей у відповідному візлі. Визначаємо новий ГКО для відображення попереджень у відповідних вузлах. Для того, щоб стежити за розвитком атаки, необхідно відстежувати IP адреси джерела і діяльність атаки. ГКО – є кортежем $G_{ГКО} = (A, E, P)$. Множина A містить усі оповіщення. Оповіщення, $a \in A$ є структурою даних, що представляють IP адресу джерела, IP адресу одержувача, тип оповіщення, а також часову мітку. Кожне повідомлення відноситься до пари вершин (v_c; v_d) в СГА, використовуючи функцію *map(a)*, тобто відображає:

$$map(a): a \rightarrow \{(v_c, v_d) | (a.дж \in v_c.вузол) \wedge (a.пр \in v_d.\ вузол) \wedge (a.кл = v_c.враз)\} \quad (2.3)$$

де v_c – вершина, яка відображає вразливість; v_d - вершина, яка відображає використання вразливості; *а.дж* – оповіщення з IP адресою джерела; $v_c.вузол$ – вершина, яка відповідає конкретному вузлу в хмарному середовищі; *а.пр* – оповіщення з IP адресою вузла призначення; $v_d.\ вузол$ – вершина, яка відповідає

конкретному вузлу в хмарному середовищі, пов’язаному з $v_c.вузол;$ $a.кл$ – оповіщення з класом вразливості; $v_c.враз$ – вразливість у вузлі, який розглядається.

Спрямовані ребра, представляють собою кореляцію між двома попередженнями ($a,\ a'$), у випадку коли критерії, що приведені нижче, будуть виконані:

$$(a.чс < a'.чс) \wedge (a'.чс - a.чс < \text{порогове значення}), \quad (2.4)$$

де $a.чс$ – оповіщення з часовою міткою в попередньому вузлі; $a'.чс$ – оповіщення з часовою міткою в наступному вузлі.

$$\exists\ (v_c, v_d) \in E_{pre} : (a.пр \in v_d\ .вузол \wedge a'.дж \in v_c.вузол). \quad (2.5)$$

ГКОвключає в себе P – набір маршрутів проходження атаки. Маршрут $S_i \subset P$ представляє собою набір пов'язаних оповіщень в хронологічному порядку, які належать одному і тому ж сценарію атаки.

Пропустимо, що A містить агреговані оповіщення, а не пусті. Необроблені сигнали, що мають те ж джерело адреси відправника і одержувача, тип атаки і мітку часу в межах заданого вікна, які агрегуються як «агреговані оповіщення». Кожна упорядкована пара $(a,\ a')$ в ГКО зіставляється з двома сусідніми вершинами в СГА з різницею часових міток двох попереджень, протягом заздалегідь визначеного граничного значення. ГКО показує залежність попереджень в хронологічному порядку і ми можемо знайти попередження, пов'язані в тому ж сценарії атаки шляхом пошуку оповіщення в ГКО. Безліч набірів маршрутів проходження атаки в графі кореляції оповіщення використовуються для зберігання всіх шляхів від кореня графу до цільового вузла в СГА, і кожен шлях $Si \subset P$ являється оповіщенням, які належать одному і тому ж сценарію атаки.

Розроблений алгоритм аналізу загроз у ВХС представлено на рис. 2.5.

Пояснимо спосіб використання СГА і ГКО, щоб передбачити поведінку зловмисника. Алгоритм кореляції оповіщенння (див. рис. 2.5) для кожного оповіщення виявляє і повертає до бази даних один або кілька шляхів Si. Кожне оповіщення, яке отримано від системи виявлення вторгнень, додається в ГКО, якщо воно там ще не існує. Для цього нового оповіщення, відповідна вершина в СГА

знаходиться за допомогою функції *map(a)*. Для цієї вершини в СГА, оповіщення, пов'язані з його вершинами типу $B_к$ корелюються з новими оповіщеннями. Це створює новий набір попереджень, які належать до маршрута *Si* в ГКО або створюють новий маршрут *Si+1* в *Si*. В кінці роботи алгоритму до оповыщення додається ідентифікатор мережі, щоб у наступні рази більш швидше проходити цей алгоритм. В кінці алгоритм повертає маршрут проходження атаки в ГКО.

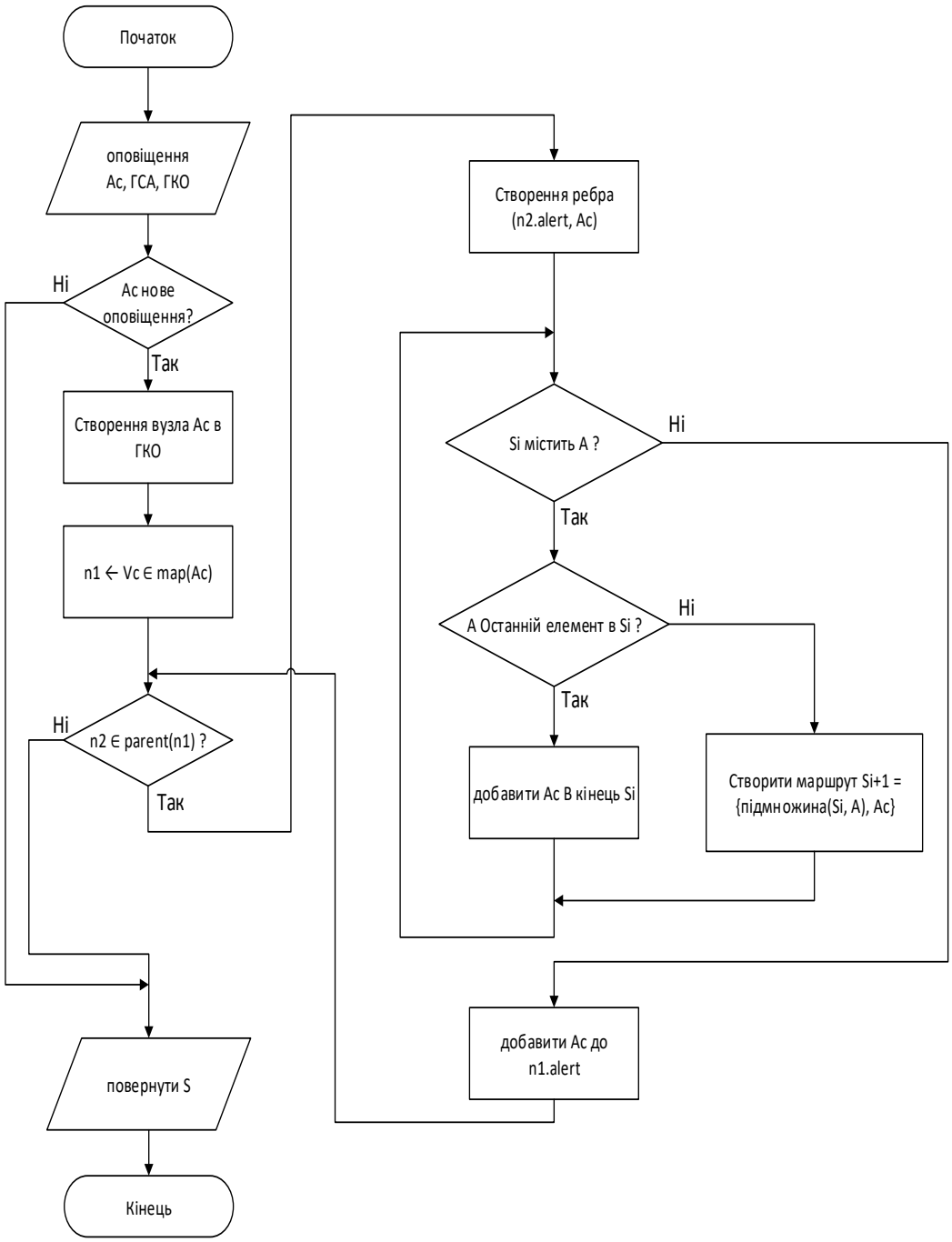

Рис. 2.5 - Алгоритм аналізу загроз у хмарному віртуальному середовищі

Для всіх віртуальних машин в мережі в залежності від кількості властивих їм вразливостей, і оцінки впливу вразливостей на віртуальне хмарне середовище назначається відповідний індекс. Оцінка впливу уразливості [42], дає можливість визначити конфіденційність, цілісність і доступність впливу уразливості.

2.4 Розробка моделі вибору контрзаходів

Необхідно провести розробку математичної моделі вибору контрзаходів. Для вибору контрзаходів для сценарію атаки. Коли уразливості виявляються або деякі віртуальні машини ідентифіковані як підозрілі, можуть бути прийняті контрзаходи, щоб обмежити можливість атакуючого. Метою контрзаходів є захист віртуальних машин від компрометації.

Використаємо граф атак в якості моделі метричної системи безпеки для оцінки ризиків безпеки [46]. Для того, щоб оцінити стан ризику мережевої безпеки та для поточної конфігурації мережі, необхідно використовувати показники безпеки в графі атак (щоб виміряти ймовірність ризику). Граф атаки містить в собі інформацію про вразливості системи. Для початкового або зовнішнього вузлів (тобто корінь графа, $B_{кор} \subseteq B_{д}$), завжди апріорна ймовірність визначається як ймовірності джерела загрози. Для позначення ймовірності апріорного ризику кореневого вузла графа в роботі використовується показник *GV* з національної бази вразливостей. Як правило значення *GV* присвоюється висока ймовірність, наприклад, від 0,7 до 1.

Для кожного вузла кожний крок атаки $E \in B_{к}$ матиме можливість бути експлуатованим; уразливості позначається як *GM*[*e*]. GM[*e*] призначається відповідно до базового показником $H_{(v)}$ від CVSS (Common Vulnerability Scoring System). Базова оцінка, як показано у формулі (2.6) [42], обчислюється коефіцієнтом впливу і використанням уразливості. Базова оцінка може бути одержана безпосередньо з національної бази даних вразливостей [44] шляхом пошуку уразливостей завдяки їх CVE ID.

$$H_{(v)}=(0,6\ I_v+0,4E-1,5)f(I_v), \qquad (2.6)$$

де, I_v = 10.41(1−(1−C)(1−I)(1−A)), E=20$A_cA_uA_v$,

$$f(I_v)=\begin{cases} 0 & \text{де } I_v=0, \\ 1{,}176 & \text{в іншому випадку.} \end{cases} \quad (2.7)$$

Величина впливу (I_v) визначається трьома основними параметрами безпеки, а саме – конфіденційністю (C), цілісністю (I) і доступністю (A). Використання вразливостей складається з вектора доступу (A_v), складності доступу (A_c) і аутентифікації (A_u). Значення $H_{(v)}$ може змінюватися в діапазоні від 0 до 10. У графі атак кожен внутрішній вузол відноситься до значення $H_{(v)}$ значення поділене на 10

$$GM[e] = H_{(v)}/10, \forall e \in B_к \quad (2.8)$$

У графі атаки відношення між вершинами можуть бути диз'юнктивними або кон'юнктівні відповідно з тим, як вони пов'язані між собою через умови їх залежностей [47]. Такі відношення можуть бути представлені у вигляді умовної ймовірності.

Застосуємо стратегії пом'якшення наслідків відповідно до отриманих попереджень про вразливості. Для забезпечення захисту віртуальних хмарних ресурсів необхідно зберігати усі відомі інструкції по контраходам в базі даних контрзаходів.

$$Q = \{Z_1, Z_2, ..., Z_n\}, \quad (2.9)$$

де Q – база даних контрзаходів; Z – контрзахід.

Кожен $Z \in Q$ і є кортежем $Z = (B, I)$, де B – це вартість, тобто витрати, необхідні для застосування контрзаходів з точки зору ресурсів і оперативної складності, і визначається в діапазоні від 1 до 5 і вище, де найвищий бал означає більш високу вартість; I – інтрузивність, що визначає негативний ефект, який призводить контрзахід до «Угоди про рівень обслуговування» (SLA). Наприклад, якщо контрзахід немає впливу на SLA, то значення інтрузивності дорівнює 0.

Загалом, в базі даних контрзаходів може бути багата кількість контрзаходів, які можуть бути застосовані до хмарного віртуального середовища, в залежності від наявних методів, які можна використовувати для реалізайії контрзаходів в віртуальному хмарному середовищу. Кілька загальних контрзаходів перераховані

в таблиці 2.1. Оптимальний вибір контрзахідів є багатоцільовим завданням оптимізації, для розрахунку минимального значення втручання – MIN (інтрузивність, вартість) та MAX (вигода).

Таблиця 2.1

Можливі типи контрзаходів

No.	*Z*	*I*	*B*
1	Перенаправлення трафіку	3	3
2	Ізоляція трафіку	4	2
3	Глибока інспекція пакетів	3	3
4	Створення правил фільтрації	1	2
5	Зміна MAC-адресу	2	1
6	Зміна IP адреси	2	1
7	Блок-порт	4	1
8	Патч програмного забезпечення	5	4
9	Карантин	5	2
10	Зміна конфігурації мережі	0	5
11	Зміна мережевої топологія	0	5

Контрзахід, який дає найменше значення комплексного показника вибору контрміри визначається як потрібний. Сценарій графа атаки та граф кореляції оповіщення також оновлюються перед завершенням алгоритму.

Для досягнення мети вибору контрміри запропоновано комплексний показник вибору контрміри *Y*.

$$Y= [(k_1 I)+(k_2 B)]/n, \quad (2.10)$$

де k_1, k_2 – характеристики впливу показників контрміри; *I* – інтрузивність, яка відображає величину негативного ефекту, який призводить контрзахід; *B* – вартість, яка відображає витрати, які необхідні для застосування контрзаходів з точки зору ресурсів і оперативної складності (чим вище показник тим більша вартість); *n* – коефіцієнт нормування.

Стратегії зміни конфігурації мережі в основному можна розділити на два рівні: рівень 2 і рівень 3. На рівні 2, віртуальні мости (в тому числі тунелі, які можуть бути встановлені між двома мостами) і віртуальні локальні мережі є

основними компонентами в віртуальній мережевій хмарі системи для підключення двох віртуальних машин. Віртуальний міст є об'єктом, який надає віртуальні інтерфейси (VIFs). Віртуальні машини на різних ізольованих мостах знаходяться на другому рівні моделі взаємодії відкритих систем. VIFs на тому ж віртуальному мосту з різними мітками VLAN не можуть взаємодіяти один з одним. На основі цього методика забезпечення захисту віртуального хмарного середовища може розгорнути на другому рівні моделі OSI потрібну конфігурацію мережі для ізоляції підозрілих віртуальних машин. Наприклад, через уразливість до Arp Spoofing [48] атака не представляється можливою, коли підозріла віртуальна машина ізольована на іншому мості. В результаті, цей контрзахід роз'єднує шлях атаки в графі, змушуючи атакуючого досліджувати альтернативний шлях атаки. На третьому рівні моделі OSI можлива перебудова топології мережі, тобто інший спосіб перервати шлях атаки. За допомогою мережевого контролера, таблиці потоку на кожному OVS або OFS можуть бути модифіковані, щоб змінити топологію мережі.

З використанням підходу реконфігурації віртуальної мережі додатки верхнього рівня моделі OSI будуть відчувати мінімальний вплив від змін на нижньому рівні. Зокрема, такий підхід можливий тільки при використанні спеціалізованного ПЗ в комутаторах мережі з автоматизацією перенастроювання їх в динамічні мережі з використанням протоколу Openflow.

Контрзаходи, такі як ізоляція трафіку, можуть бути реалізовані шляхом використання інженерних можливостей OVS і OFS, щоб обмежити потенціал атаки та налаштувати віртуальну мережу для подальшої ізоляції потоку з підозрілим трафіком. Коли виконуються підозрілі дії, такі як сканування портів мережі, важливо визначити чи зловмисники активні. Наприклад, зловмисники можуть навмисно приховувати свої дії за рахунок сканування портів, щоб запобігти NIDS від виявлення їх дій. У такій ситуації, зміна конфігурації мережі змусить зловмисника виконувати більше дій по дослідженю мережі, і в свою чергу, покаже що він активний.

Для оцінки ступеню захисту віртуальної машини використаємо показник

$$F_{вм} = (O_{вм} + O_{Авм})/2, \quad (2.11)$$

де, $O_{\text{вм}}$ – оцінка вразливості віртуальної машини; $O_{\text{Авм}}$ – оцінка використання вразливості віртуальної машини.

Оцінка вразливості віртуальної машини представляє собою середній базовий бал з множини вразливостей у віртуальній машині

$$O_{\text{вм}} = \min \{10, \ln \sum e^{H(v)}\}, \quad (2.12)$$

де $H_{(v)}$ – середній базовий бал з кожної вразливості конкретної віртуальної машини; v – конкретна віртуальна машина.

Оцінка використання вразливості віртуальної машини є

$$O_{\text{Авм}} = (\min \{10, \ln \sum e^{J(v)}\})(N_{(v)}/Z_{(v)}), \quad (2.13)$$

де $J_{(v)}$ – середній бал використання вразливості з кожної вразливості конкретної віртуальної машини; $N_{(v)}$ – кількість послуг, які надаються віртуальною машиною; $Z_{(v)}$ – кількість послуг, які можуть бути надані віртуальній машині.

Тобто, $O_{\text{вм}}$ враховує базові оцінки всіх вразливостей на віртуальній машині. Кожна базова оцінка вразливостей враховує ступінь можливостей атакуючого використовувати цю вразливість, а також величину пошкоджень, яку він може нанести. Експонентна сума базових показників дозволяє оцінити відхилення їх значень в логарифмічному масштабі на основі кількості вразливостей.

Оцінка вразливості віртуальної машини $O_{\text{Авм}}$ з іншого боку, відображає спроможність нападника до використання вразливості віртуальної машини і залежить від співвідношення кількості використаних мережевих послуг до загального числа можливих мережевих послуг. Висока оцінка $F_{\text{вм}}$ означає, що у вразливостей існує велика кількість можливих шляхів для досягнення мети зловмисника.

Таким чином $F_{\text{вм}}$ представляє собою кількісну оцінку рівня захисту кожної віртуальної машини у віртуальній хмарній системі. Для того, щоб запобігти нападам від використання інших вразливих віртуальних машин, до віртуальних машин з більш високими значеннями $F_{\text{вм}}$ необхідно застосовувати вищі ступені захисту. Для зменшення значення $F_{\text{вм}}$ обґрунтовані стратегії пом'якшення впливу атаки, в залежності від величини негативного впливу.

В основному, оцінка уразливості враховує базові оцінки всіх вразливостей на віртуальній машині. Базова оцінка показує ступінь складності для атакуючого використовувати цю вразливість і величину пошкоджень, які віртуальна машина та система в цілому може понести. Експонентне додавання базових показників дозволяє оцінити уразливості більш високих значень і збільшити оцінку в логарифмічному масштабі на основі кількості вразливостей. Використання вразливостей з іншого боку, відображає наявність віртуальної машини, на яку націлена атака і залежить від співвідношення кількості мережевих послуг які надає віртуальна машина до числа мережевих послуг які може прийняти віртуальна машина [51]. Більш висока оцінка показує більшу кількість вразливостей, а також можливі шляхи для досягнення мети зловмисника.

Показник $F_{\text{вм}}$ може бути використаний як індикатор для демонстрації стану захисту віртуальної машини, тобто віртуальна машина з більш високим значенням $F_{\text{вм}}$ означає, що може легше піддатися нападу. Для того, щоб запобігти нападам від використання інших вразливих віртуальних машин, віртуальні машини з більш високими значеннями $F_{\text{вм}}$ необхідно частіше перевіряти та застосовувати до них відповідні контрзаходи для зменшення значення $F_{\text{вм}}$.

Для кожного реалізованого віртуального хмарного середовища в якості критерію захисту обгрунтовується значення показника, при досягнені якого віртуальне хмарне середовище не зможе забезпечити задані вимоги щодо його компроментації.

Алгоритм вибору контрміри (див. рис. 2.6) показує, як вибрати оптимальні контрзаходи для даного сценарію атаки. Вхідними даними алгоритму являються оповіщення, граф атаки, а також база даних контрзаходів. Він починається з вибору вузла, який відповідає оповіщенню від мережевого агента. Перед тим як вибрати контрзаходи, ми розраховуємо відстань до вузла на який націлена атака. Якщо відстань більше, ніж порогове значення, то не здійснюється вибір контрзаходів, але оновлюється ГКО та продовжується відстеження оповіщення в системі.

Оскільки попередження генерується тільки після того, як зловмисник виконав дію, ми встановлюємо ймовірність отримання оповіщення рівну 1 і

розраховуємо вірогідність для всіх його дочірніх (вниз за графом) вузлів. Потім для всіх обраних вузлів розраховуються комплексні показники вибору контрміри, та порівнюються і приймаються. Зміна ймовірності цільового режиму дає перевагу для застосовування контрзаходів. В кінці алгоритму СГА і ГКО також будуть оновлені перед завершенням алгоритму.

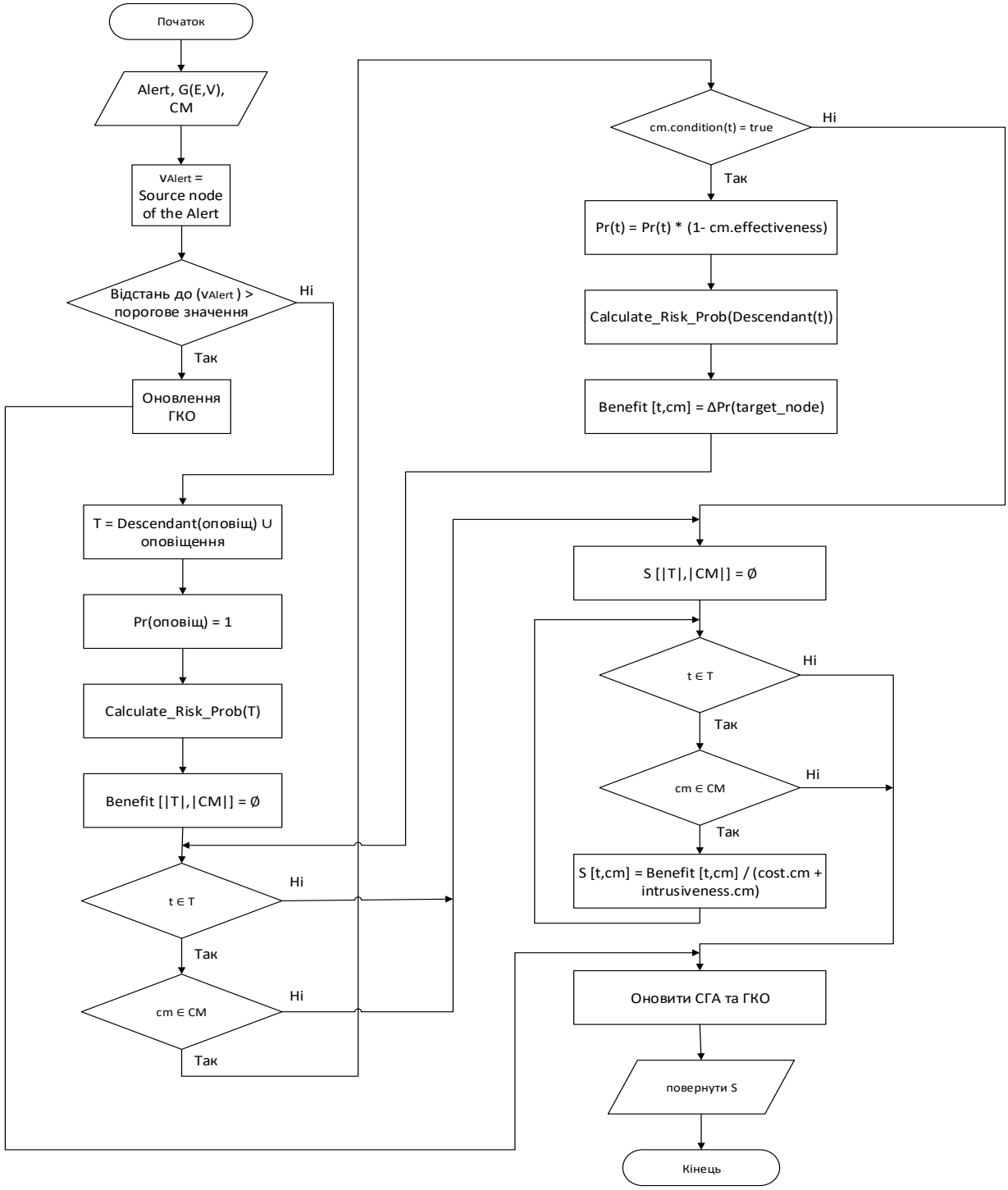

Рис. 2.6 - Алгоритм вибору контрміри

2.5 Висновки до розділу 2

1. У другому розділі проведено розробку математичних моделей для забезпечення захисту віртуальних хмарних ресурсів, а саме: математичну модель впливу атаки на віртуальні хмарні ресурси, математичну модель оцінки стану віртуальних хмарних ресурсів, математичну модель вибору контрміри на основі комплексного показника.

2. Розроблено граф атак на віртуальне хмарне середовище. Даний граф дозволяє отримати інформацію про всі відомі вразливості системи, а також показує в режимі реального часу стан на основі програмно-конфігурованих мереж системи, що дає можливість спрогнозувати можливі загрози і атаки шляхом кореляції виявлених подій.

3. Розроблено алгоритм аналізу загроз у хмарному середовищі. Результатом роботи даного алгоритму є отримання одного або декількох шляхів проходження атаки на віртуальне хмарне середовище.

4. Розроблено алгоритм вибору контрміри. Даний алгоритм показує, яким чином вибрати контрзаходи для конкретного сценарію атаки. При виконанні розробленого алгоритму враховуються показники експлуатаційної вартості виконання контрзаходу та інтрузивність, що в свою чергу вказує на рівень втручання обраного контрзаходу до змін в угоді про рівень обслуговування.

3 МЕТОДИКА ЗАХИСТУ ВІРТУАЛЬНОГО ХМАРНОГО СЕРЕДОВИЩА

Методика забезпечення захисту віртуального хмарного середовища призначена для рішеня завдання підвищення стійкості до компроментації віртуальних машин на базі виявлення атак та реконфігурації віртуальних мереж в віртуальному хмарному середовищі. Розроблена технічна методика може бути впроваджена в хмарній системі IaaS. Якщо цю методику застосувати для Cloud Service Provider (CSP), це приведе до підвищення захисту організації його роботи. Також припускаємо, що користувачі хмарних сервісів можуть вільно встановлювати будь-які операційні системи або додатки.

3.1 Побудова віртуального хмарного середовища

Методика захисту ВХС – це структура всередині кластера серверів одного ВХС. Основні компоненти цієї структури розподілені між собой на програмному рівні. База даних профілів віртуальних машин, мережевий контроллер, аналізатор атаки розташовані в одному фізичному централізованому місці – центрі управління, підключеному до програмних комутаторів на кожному сервері хмари (рис. 3.1). Мережевий агент є ПЗ реалізованим в кожному сервері хмарного середовища, яке підключено до центру управління через спеціальний окремий ізольований захищений канал, який відділений від звичайних каналів зв’язку з використанням OpenFlow tunneling та VLAN. Мережевий контролер відповідає за розгортання та впровадження контрзаходів, що грунтується на рішеннях, прийнятих аналізатором атаки, як відповідь на компроментуючі дії нападників.

Робота методики грунтується на технології віртуалізації Xen. Мережевий агент є механізмом виявлення вторгнень, які можуть бути встановлені в будь-якому Dom0 або DomU на сервері хмари Xen для збору і фільтрації повідомлень шкідливого трафіку. Виявлені вторгненя у вигляді оповіщень направляються в центр управління, коли підозрілий або аномальний трафік був виявлений. Отримавши повідомлення про загрозу, аналізатор атаки оцінює цю загрозу,

використовуючи данні з графу атаки. Далі визначається, на основі математичної моделі вибору контрміри в якості показника на основі програмно-конфігурованих мереж, які контрзаходи вибрати для пом'якшення впливу атаки. Настним кроком ініціюється передача обраних контрзаходів до мережевого контролера, де вони приймаються до реалізації в віртуальній мережі. Граф атак будується відповідно до інформації, що отримана про уразливість як в автономному режимі, так і в режимі реального часу. Автономне сканування може бути проведено шляхом виконання тестів на проникнення і сканування вразливостей за допомогою відповідного ПЗ. В режимі реального часу автономне сканування може бути викликано мережевим контролером. Після того, як нові уразливості виявляються в системі або коли контрзаходи були прийняті граф атак буде реконструйовано.

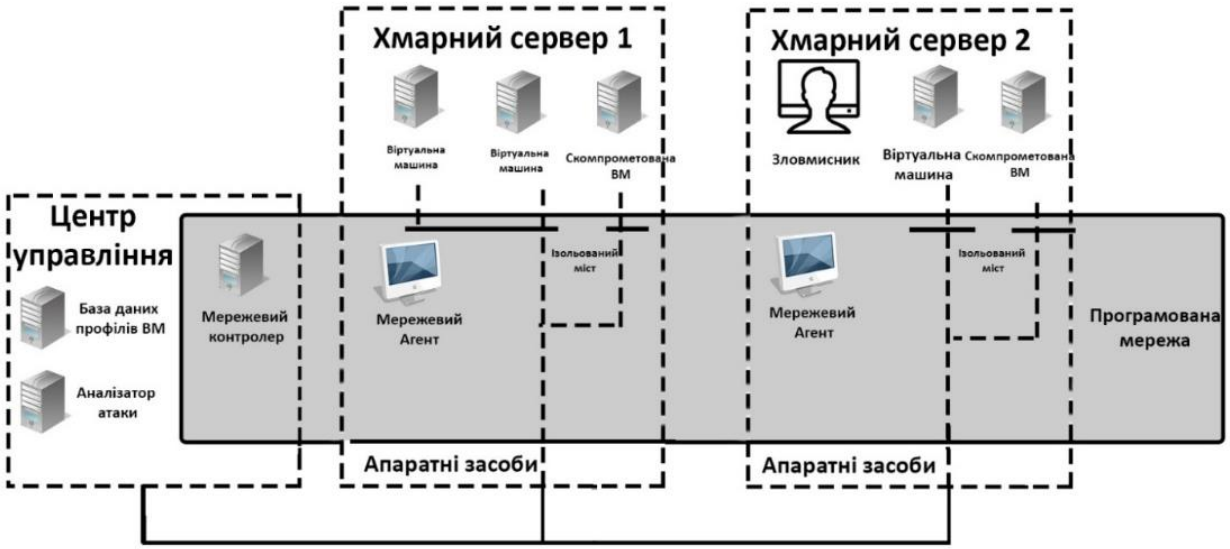

Рис. 3.1.- Схема віртуального хмарного середовища

Розглянемо елементи схеми віртуального хмарного середовища .

Мережевий агент віртуального хмарного середовища виконує сканування трафіку, що проходить через мережеві мости, які контролюють весь трафік між віртуальними машинами в фізичному хмарному сервері. Мережевий агент на основі «мережевої системи виявлення вторгнень» (NIDS) встановлюється в будь-якому Dom0 або DomU в кожному сервері хмари. Він сканує трафік, що проходить через Linux мости, які

контролюють весь трафік між віртуальними машинами в фізичному сервері хмарни. Використовується програмне забезпечення «Snort» для реалізації необхідних функцій мережевого агента в Dom0. Він перехоплює дзеркальне відображення порту на кожному віртуальному мосту в OVS. Кожен міст утворює ізольовану підмережу у віртуальній мережі і з'єднується з усіма пов'язаними віртуальними машинами. Трафік, що генерується з віртуальних машин на дзеркальному програмному забезпечені моста, буде дзеркальний до певного порту на певному мості з використанням методів SPAN, RSPAN і ERSPAN. Dom0 в середовищі Xen є областю привілегій, яка включає в себе віртуальний комутатор для комутації трафіку між мережевими і віртуальними драйверами віртуальних машин для фізичного мережевого інтерфейсу сервера хмари. Сканування трафіку в Dom0 більш ефективне, так як весь трафік в сервері хмари повинен пройти через нього, не залежно від встановлених віртуальних машин.

База даних профілів віртуальних машин містить інформацію про стан кожної віртуальної машини. Віртуальні машини в хмарному середовищу профілюються таким чином, щоб отримати точну інформацію про їх стан, стан служб, що виконуються на них, стан відкритих портів і т.д. Будь-яка віртуальна машина, яка підключена до більшої кількості машин є більш важливоютому, що необхідно зберігати інформацію про служби, що працюють на віртуальній машині для того щоб перевірити достовірність попереджень, що відносяться до цієї віртуальної машини. Зловмисник може використовувати програму сканування портів для виконання інтенсивного дослідження мережі та для пошуку відкритих портів на будь-якій віртуальній машині. Тому інформація про будь-який відкритий порт на віртуальній машині та історія відкритих портів грає істотну роль у визначенні наскільки вразливою є віртуальна машина. Всі ці фактори в сукупності створюють профіль віртуальної машини. Профілі віртуальних машин зберігаються в базі даних і містять інформацію про уразливість, попередження від мережевого аналізатора, відкриті порти, сервіси.

Профілі ВМ зберігаються в єдиній базі даних. Такі дані надходять від:

- графа атак: при генерації графа атак, кожна виявлена уразливість, додається до відповідної ВМ в базі даних;
- мережевого агенту: він записує попередження за участю ВМ в базі даних профілів ВМ.
- мережевого контролеру: він записує шаблони трафіку, які пов'язані з віртуальною машиною, засновані на 5 параметрах (MAC-адреса відправника, MAC-адреса призначення, IP адреса джерела, IP адреса призначення). Система може мати шаблони трафіку, де пакети виходять від однієї IP адреси і доставляються до кількох IP адрес призначення, і навпаки.

Мережевий контролер віртуального хмарного середовища реалізує віртуальну функцію реконфігурації мережі на основі протоколу openflow. Використовуються інтегровані функції управління для openvswitch і openflow switch в мережевому контролері, який дозволяє ВХС встановлювати правила безпеки або фільтрації на комплексній основі.

Мережевий контролер збирає інформацію про мережу, поточні мережі openflow і забезпечує їх введення в аналізатор атаки, для побудови графів атак. Інформація про зміну топології автоматично відправляється на контролер, а потім переробляється для відновлення графу атак. Мережевий контролер також відповідає за застосування контрзаходів, отриманих від аналізатора атак. Через внутрішні модулі виявлення, які використовують DNS, DHCP, LLDP і flow-initiations [45], мережевий контролер здатний виявити інформацію про підключення до мережі від OVS і OFS. Ця інформація включає в себе поточні канали передачі даних на кожному комутаторі і детальну інформацію про потіки та пов'язані з цим заголовки пакетів, такі як TCP/IP і MAC. Зміни проходження потоку і інформація про зміну топології, будуть автоматично відправлені до контролера, а потім доставляються в аналізатор атаки для відновлення графу атак.

Функцією мережевого контролера є надання допомоги модулю аналізатора атаки. Відповідно до протоколу OpenFlow [39], коли контролер отримує перший пакет потоку, він зберігає пакет і перевіряє таблицю потоку для дотримання політики трафіку. Після того, як транспортний потік підтверджується на

мережевому контроллері. Усі пакети потоку не обробляються мережевим контролером, але контролюються мережевим агентом на наявність аномального трафіку. Він також відповідає за застосування контрзаходів отриманих від аналізатора атаки. На основі методики захисту ВХС обираються і виконуються відповідні контрзаходи для виконання мережевим контролером. Якщо отримується оповіщення, якє ідентифікується відповідно до відомих атак, або ВМ визначається як скомпроментована, мережевий контролер негайно ізолює віртуальну машину. Оповіщення з середнім рівнем загрози переводить ВМ в клас підозрілих. Контрзаходи в такому випадку переводять підозрілу ВМ з стану експлуатації в режим карантину, а також переводять мережевий агент до режиму глибокого аналізу пакетів (DPI). Оповіщення з невеликим рівнем загрози можуть бути згенеровані через наявність вразливої ВМ, але відносяться до іншої віртуальної машини з цієї мережі. Для цього випадку, щоб перехопити нормальний трафік ВМ, підозрілий трафік ВМ буде переведений в режим перевірки. Будуть вжиті заходи, такі як обмеження пропускної здатності потоку або зміна конфігурації мережі.

Аналізатор атаки виконує основні функції методики захисту віртуального хмарного середовища, який включає в себе такі процедури, як побудова і оновлення сценація графа атак та графа кореляції оповіщення, а також оцінка та вибір контрзаходів.

Процес побудови і використання сценарія графа атак складається з таких етапів: збір інформації, будівництво графа атак. За допомогою цієї інформації, шляхи атаки можуть бути змодельовані за допомогою СГА. Кожен вузол графа атак представляє можливість до використання вразливості. Кожен шлях від початкового вузла до вузла мети зловмисника являє собою успішну атаку.

Таким чином, граф атак методики забезпечення захисту віртуального хмарного середовища на основі програмно-конфігурованих мереж розробляють на основі наступної інформації:

- інформація про систему в ВХС збирається з контролера вузла. Інформація включає в себе число ВМ в сервері хмари, служби, які виконуються на кожній віртуальній машині, і інформацію про віртуальні інтерфейси (VIF) з ВМ;

- топологія і інформація про конфігурацію ВХС збирається з мережевого контролера, яка включає в себе віртуальну топологію мережі, підключенні хости, мережеві підключення до ВМ, IP адресу кожної ВМ, MAC-адресу кожного мережевого інтерфейсу ВМ, а також інформацію про потік трафіку;

- відомості про уразливість ВХС генеруються на вимогу сканування вразливостей, яке ініційоване мережевими контролером і агентом. Використовується регулярне тестування на проникнення з використанням баз даних вразливостей: Open Source Vulnerability Database [[41], NIST National Vulnerability Database (NVD) [44].

Аналізатор атаки також визначає кореляцію та повідомляє про проведений аналіз операцій. Цей компонент виконує дві основні функції:

- конструює оповіщення кореляції графа (ГКО);
- надає інформацію про погрози і відповідні заходи протидії для мережевого контролера для віртуальної зміни конфігурації мережі.

Отримавши повідомлення від мережевого агенту, аналізатор відповідає попередженням в граф кореляції оповіщення. Якщо попередження вже існує в графі і це відома атака, аналізатор атаки виконує процедуру вибору контрзаходів відповідно до алгоритму, що приведений на рис. 2.7, а потім дає команду мережевому контролеру негайно розгорнути контрзаходи або пом'якшити дії.

Якщо попередження є новим, аналізатор атаки буде генерувати оповіщення кореляції і аналіз відповідно до алгоритму, що приведений (рис. 2.5), а також оновлень ГКО і СГА. Цей алгоритм корелює кожен новий сигнал до відповідного оповіщення про набір кореляції (тобто в тому ж сценарії атаки). Обраний контрзахід застосовується мережевим контролером на основі результатів оцінки. Якщо система попереджується про нову уразливість і такої немає в базі даних методики забезпечення захисту віртуального хмарного середовища аналізатор атаки додає його до атакуючого графа, а потім відтворює його.

3.2 Методика захисту віртуального хмарного середовища

Розробимо методику захисту віртуального хмарного середовища мереж, яка базується на основі виявлення атак та реконфігурації віртуальних мереж, а також підвищує захист віртуальних машин до компрометації, використовуючи комплексний показник [7, 8].

В розгорнутому вигляді методику представлено алгоритмом на рис. 3.2.

Етапи методики забезпечення захисту віртуального хмарного середовища:

1. Отримується оповіщення про загрозу.
2. Проводиться порівняння цього оповіщення з уже відомими в системі.
3. Проводиться кореляція оповіщення, на основі якої відбувається оновлення графа кореляції та розрахунок показника захисту віртуальної машини.
4. Вибірається контрміра на основі значення комплексного показника вибору контрміри.
5. Відправлення вибраної контрміри до мережевого контролеру.

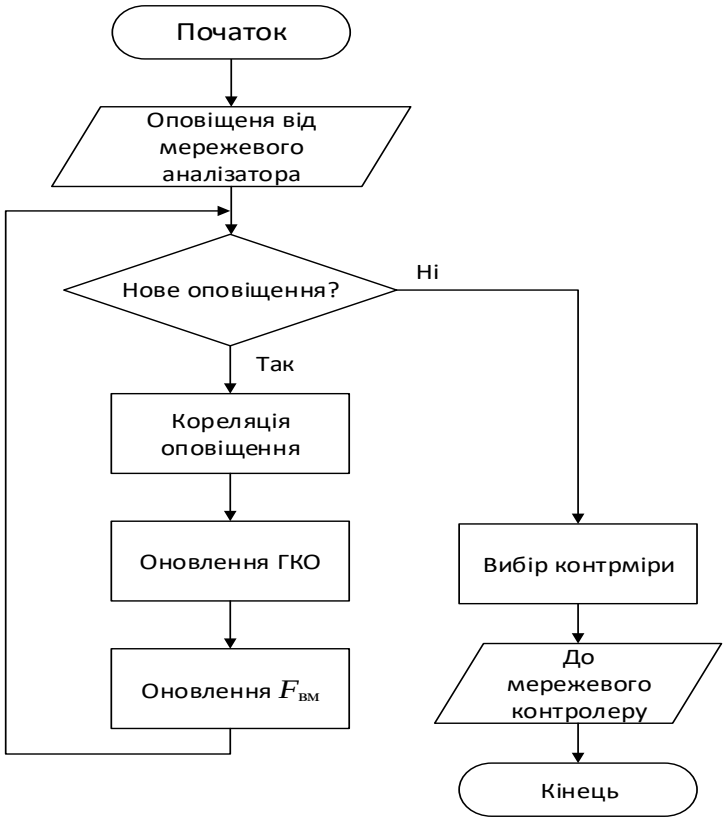

Рис. 3.2. Алгоритм забезпечення захисту віртуального хмарного середовища

Логіка алгоритму полягає у наступному: спочатку отримуємо оповіщення про загрозу, потім відбувається порівняння цього оповіщення з вже відомими в системі; наступним кроком є кореляція оповіщення, тоді відбувається оновлення графа кореляції та розрахунок показника надійності ВМ. Після чього можно зробити вибір контрміри. І завершальним кроком алгоритму є відправлення вибраної контрміри до контролеру.

Методика використовує можливість програмування віртуальних мереж, що дозволяє системі побудувати динамічну систему протидиї вторгнень. На основі методу дзеркального відображення трафіку [18], методика створює повну копію трафіку, щоб звести до мінімуму затримки в мережі при скануванні трафіку користувачів в порівнянні з традиційним «врізанням в провід». Вона дозволяє в хмарі встановлювати режими контролю і карантину підозрілих віртуальних машин відповідно до їх поточного стану уразливості в поточному СГА. На основі колективної поведінки віртуальних машин в СГА, методика забезпечує проведення відповідних заходів (наприклад DPI або фільтрації трафіку на підозрілих віртуальних машинах). Використовуючи цей підхід, методика дає можливість не блокувати потоки трафіку підозрілої ВМ на ранній стадії атаки.

3.3 Оцінка ефективності методики захисту віртуального хмарного середовища

Необхідно оцінити продуктивність віртуального хмарного середовища та технічних експлуатаційних витрат мережі при використанні розробленої методики.

Проведемо аналіз ефективності методики. Цей аналіз потребує демонстрації виконання розробленої методики, яка покаже підвищення стійкості до компрометації віртуальних машин на основі виявлення атак та реконфігурації віртуальних мереж. Для цього було розроблено віртуальне хмарне середовище для тестування мережі, яке складається з усіх представлених компонентів.

Для оцінки продуктивності системи віртуального хмарного середовища розроблено систему віртуальної хмари (рис. 3.3), що складається з відкритих

(публічних) віртуальних серверів і приватних віртуальних машин (ВМ), де встановлюються віртуальні домени. Сервери хмари 1 і 2 підключені до мережі Інтернет через зовнішній брандмауер. У демілітаризованій зоні (DMZ) на сервері 1 є один поштовий сервер, один сервер DNS і один веб-сервер. Публічна мережа на сервері 2 складається з SQL сервера і NAT сервера. Віддалений доступ до віртуальних машин в приватній мережі управляється за допомогою SSHD (тобто SSH Daemon) з сервера шлюзу NAT.

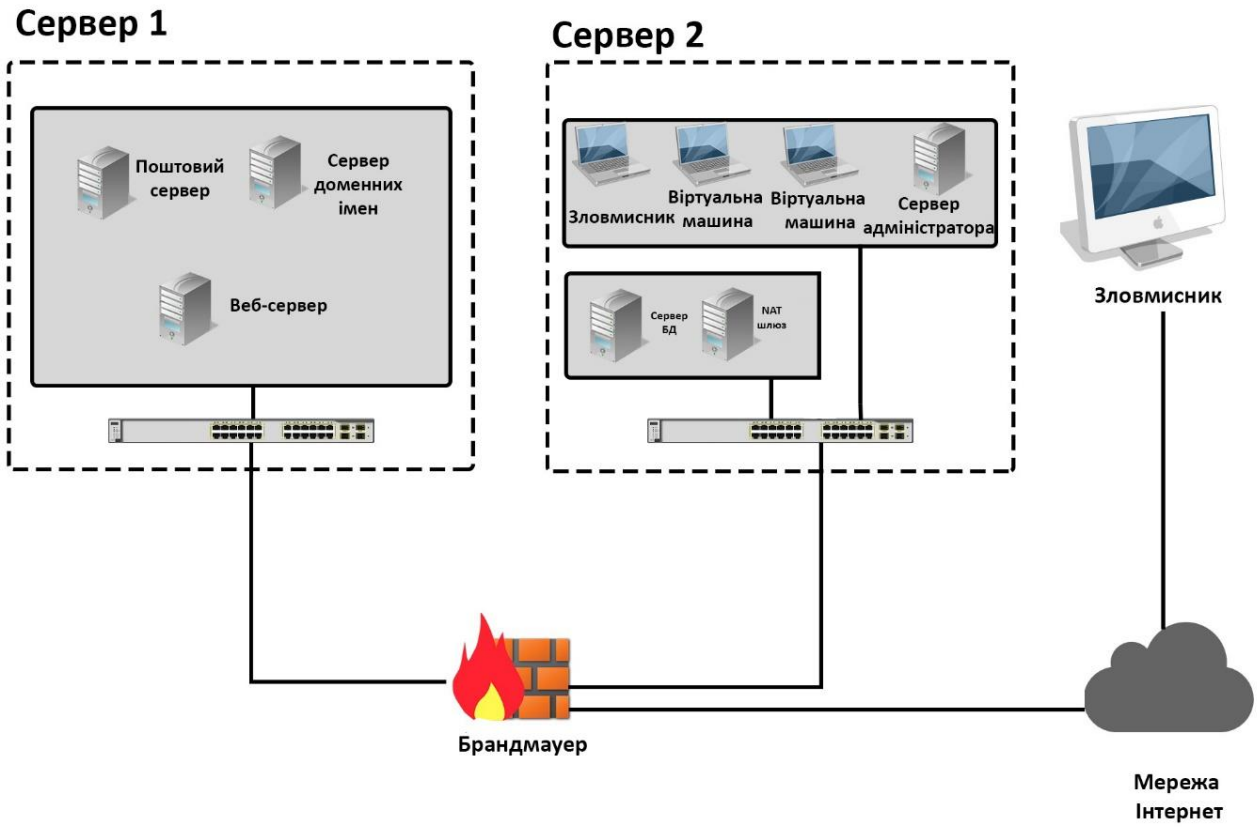

Рис. 3.3 - Структурна схема тестового стенду

При проектуванні тестового стенду для аналізу продуктивності системи ВХС враховуються різні окремі варіанти початку атаки. Для реалізації тестового стенду віртуального хмарного середовища було використано сучасну лабораторну базу Державного університету телекомунікацій. В якості першого та другого сервера хмари використовувалися сервера HPE ProLiant DL60 gen9 та комутатор 3-го рівня HP 3800-48G-PoE+-4SFP+ Switch J9574A.

Важливим етапом дослідження являється побудова сценарію графа атак. Граф може бути згенерований з використанням топології мережі та інформації про уразливості, що показано на рис. 3.4. При проведені атаки, система генерує різні попередження, які можуть бути пов'язані з вузлами в графі атак.

Створення графа атак вимагає знань про підключення до мережі, працюючих служб і інформацію про уразливості. Ця інформація надається графу атак в якості вхідних даних. Кожного разу, коли нова уразливість виявлена або є зміни в підключені до мережі, або до служб, що працюють через ці підключення, проводиться оновлення інформації яка надається графу атак і старий граф атак змінюється на новий. СГА надає інформацію про можливі шляхи, які зловмисник може використати.

Наступним важливим етапом дослідження являється побудова сценарію графа кореляції оповіщення. ГКО служить для підтвердження поведінки нападників, а також допомагає у визначенні хибних спрацьовувань і хибних негативних результатів.

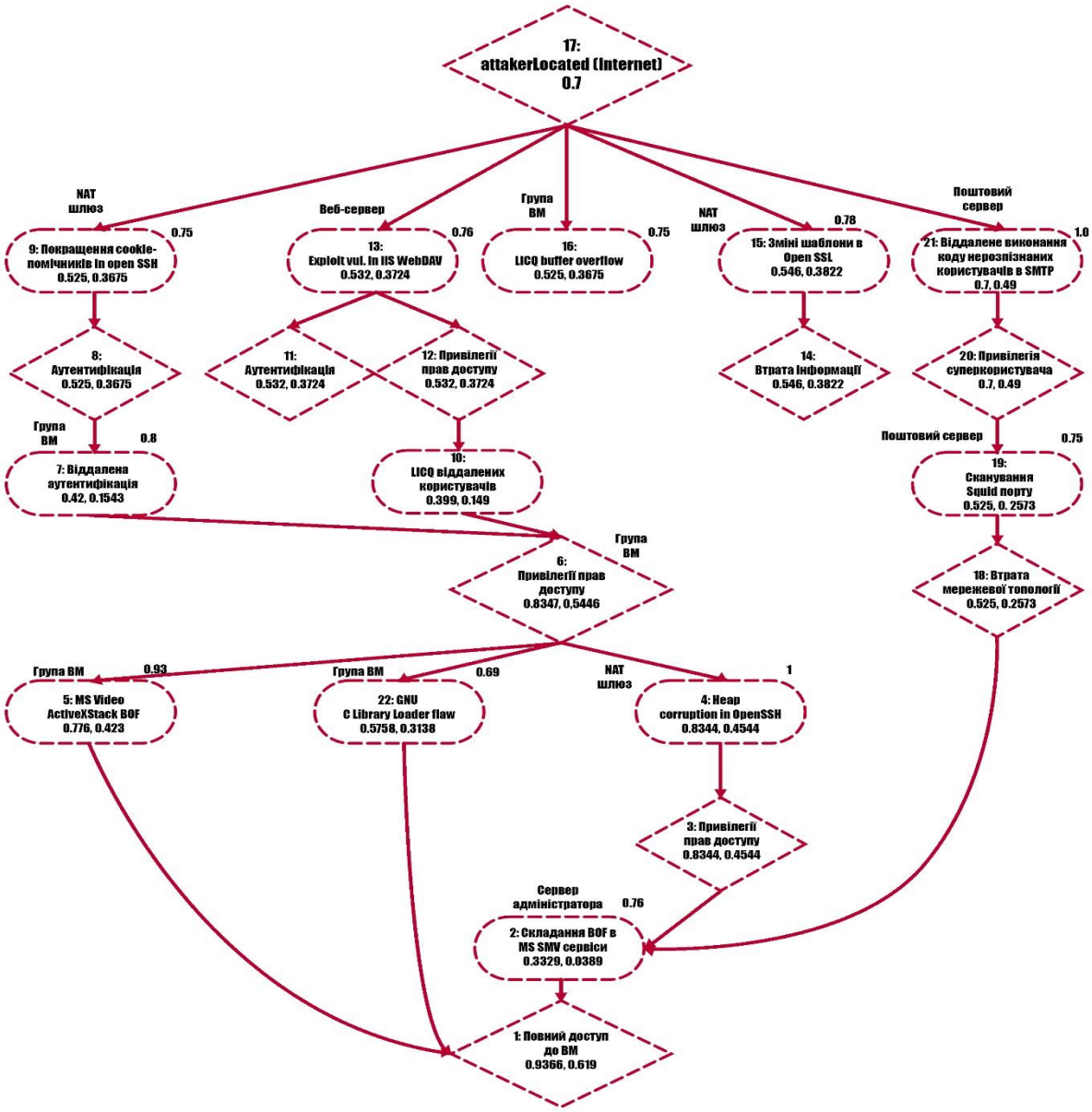

Рис. 3.4 - Граф атаки для тестової мережі.

Граф кореляції оповіщення представляє собою:

- набір вершин, що відображають вразливості;

- вершин, що відображають використання цих вразливостей;

- набір орієнтованих ребер, які звязують між собою вершини які входять до можливих шляхів проходження атаки з врахуванням часових міток, відповідності отриманих оповіщень від мережевого агенту до відповідних вузлів в системі та відповідності класу цих опоіщень до можливих вразливостей у відповідному візлі.

Граф кореляції оповіщення також може бути корисним для прогнозування подальших кроків нападника.

Для підтвердження ефективності роботи методики забезпечення захисту ВХС доцільно провести експеримент в середовищі приватної хмари.

Визначено, що метою зловмисника не завжди являється компроментація яких-небуть глобальних сервісів або ресурсів. Інколи метою можуть виступати штатні віртуальні машини. Тому проведення експерименту є доцільним, коли мета зловмисників – компроментація ВМ в приватній мережі. Для аналізу продуктивності і тесту методики забезпечення захисту віртуального хмарного середовища було розширено конфігурацію приведену на рис. 4.1, щоб створити інше тестове середовище, яке включає в себе 14 віртуальних машин, що працюють на серверах хмари та налаштовані окремо одна від одної. Цей крок являться важливим етапом для створення СГА для кожної віртуальної машини. Віртуальні машини тестового середовища приватної хмари працюють на операційних системах Windows і Linux. Використано IP адресацію: 10.10.0.0/24. Мережа містить велику кількість вразливостей, пов'язаних з операційними системами і додатками. Було створено сценарії тестування на проникнення в віртуальне приватне хмарне середовище з використанням в якості нападників програмного забезпечення Metasploit [49] і Armitage [50]. Даний підхід дозволяє емулювати атаку з різних місць у внутрішніх і зовнішніх вузлах, а також запускати різноманітні атаки на основі вразливостей в кожній віртуальній машині.

Нижче на рис. 3.5 представлено діаграму оцінки ВМ $O_{\text{вм}}$ – оцінки вразливості ВМ та $O_{\text{Авм}}$ – оцінки використання вразливості віртуальної машини.

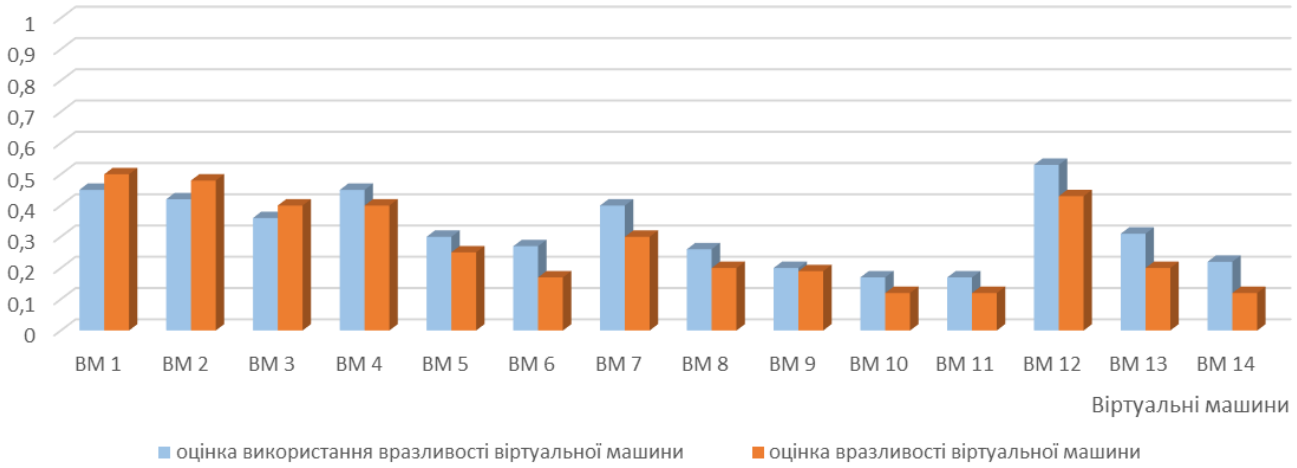

Рис - 3.5. Діаграма оцінки віртуальних

Діаграма демонструє на скільки вразливості у ВМ можуть бути використані зловмисником для їх компроментації. Зазначимо, що чим менші ці показники тим менші ризики того що зловмисник зможе досягти своєї мети. Якщо один з показників вище ніж 0,5 то з високою вірогідністю можна вважати, що загальний показник захисту системи буде також високим. Це в свою чергу означає, що існує висока імовірність її компроментації і до неї будуть прийняті відповідні контрміри. Оцінки ВМ в залежності від показника захисту пок. у вигляді діаграми на рис. 3.6.

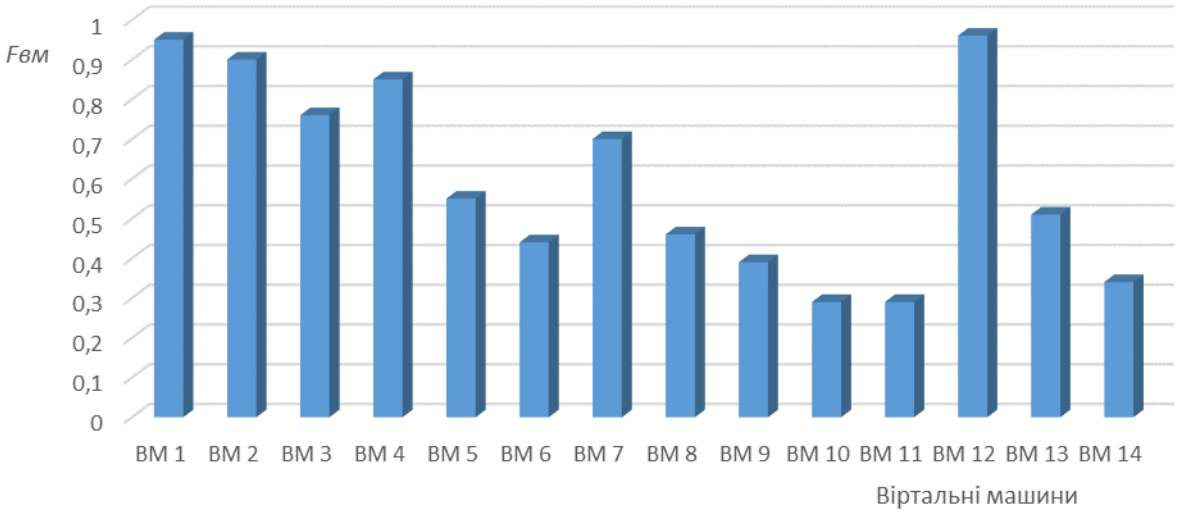

Рис. 3.6 - Діаграма оцінки віртуальних машин в залежності від показника захисту

Діаграма оцінки ВМ в залежності від показника захисту демонструє загальний стан захисту в приватному хмарному середовищі. Зазначимо, що чим

нижче значення показника захисту тим вищий стан захисту віртуальної машини у приватному хмарному середовищі.

Нижче приведено діаграму комплексного показника вибору контрміри *Y* (рис. 3.7), що відображає величину негативного впливу, який призводить контрзахід до угоди про рівень обслуговування SLA. Даний показник було запропоновано вище в роботі та представлено формулою (2.13). Також даний графік відображає витрати, які необхідні для застосування контрзаходів з точки зору ресурсів і оперативної складності (чим вище показник тим більша вартість). Дані показники разом відображають комплексний показник вибору контрміри *Y*. Слід зазначити, що контрміри з високим комплексним показником вибору контрміри *Y* відповідають більш серйозним та рідко використовуваним контрзаходам, котрі можуть істотно змінити практично всю інфраструктуру віртуального хмарного середовища.

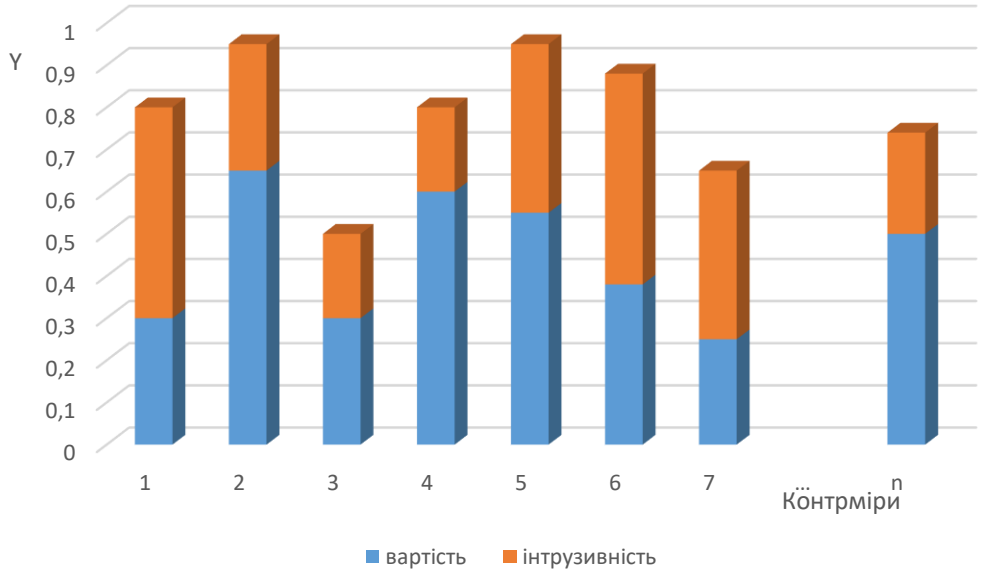

Рис. 4.7 - Діаграма комплексного показника вибору контрміри *Y*

На рис. 3.8 наведено значення показника захисту віртуальної машини для тестових віртуальних машин до прийняття контрміри та після цього.

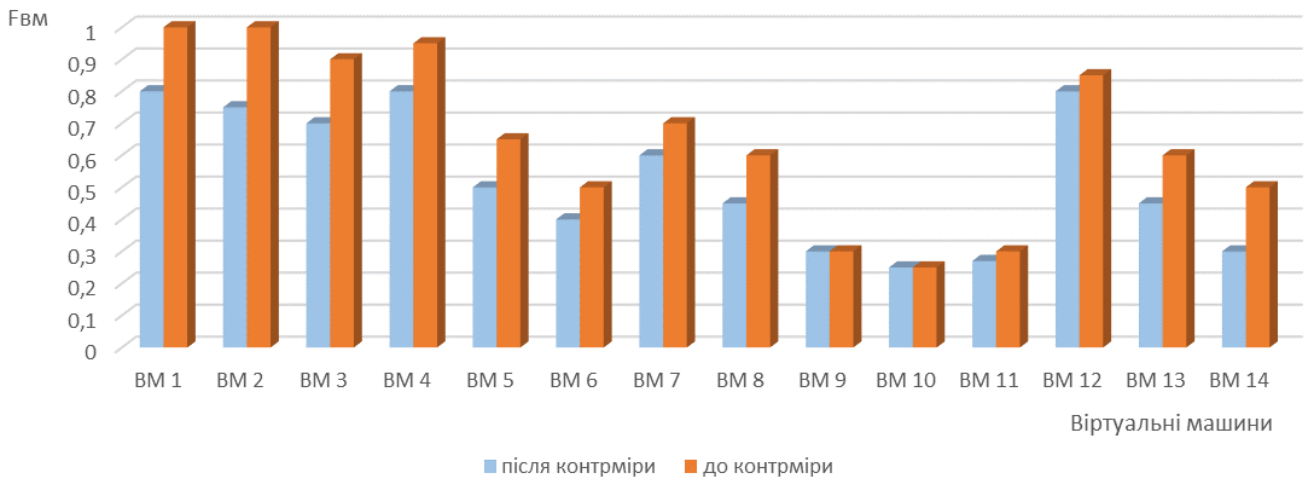

Рис. 4.8. Зміна показника захисту $F_{вм}$ віртуальної

Аналіз діаграми показує, що середній показник зниження вразливості віртуальних машин за показником $F_{вм}$ складає 8 %, що дає можливість підвищити ефективність віртуального хмарного середовища.

Чим більше об'єм хмарного середовища, що планується використовувати, тим більше, за законом великих чисел, вірогідність помилкових тривог. Система хмарного віртуального середовища з сотнями вузлів матиме величезну кількість оповіщень, що генеруються Snort. Тому необхідно розробити ефективний механізм перевірки правдивості оповіщень. Так як Snort може бути запрограмовано таким чином, щоб генерувати оповіщення з ідентифікатором CVEID, пропонується проводити перевірку на наявність вразливості в базі даних вразливостей. Якщо виконується ця умова, то існування цієї вразливості в СГА означає, що імовірно буде проводитися реальна атака. Таким чином, кожне нове хибне спрацювання не приведе до збільшення відсотку не продуктивного використання ресурсів віртуального хмарного середовища.

Не можливо корректно без хибних спрацювать обробити отовіщення з індикатором атаки «нульового дня», де уразливість виявлена зловмисником або системою, не виявляється мережевим сканером вразливостей. В такому випадку усі оповіщення, які мають ідентифікатор атаки «нульового дня» вважаються реальними і потребують внесення до графа атаки. Не дивлячись на те, що не існує

відповідного вузла в СГА. Важливо відзначити, що сканер вразливостей повинен бути в змозі виявити найостанніші уразливості і синхронізації з останньої бази даних вразливостей, щоб зменшити ймовірність атак «нульового дня».

Оцінимо вплив методики забезпечення захисту віртуального хмарного середовища на продуктивність системи. Це необхідно для того, щоб визначити виличину можливого трафіку на одному сервері та відсоток утілізації роботи центрального процесора обумовленого роботою методики в залежності від кількості мережевого трафіку для того щоб провести масштабування приватної віртуальної хмарної системи до великої публічної віртуальної хмарної системи.

Щоб продемонструвати ефективність використання методики проведено порівняльні дослідження для двох підходів реалізації паравіртуалізації.

Перший підхід грунтується на реалізації Dom0 і DomU з дзеркальним відображенням трафіку. В основі другого підхіду лежить використання додаткогового проксі-сервера для виявлення атак. У першому варіанті встановено дві віртуальні мережі в кожному сервері хмари: звичайної мережі та мережі моніторингу. Мережевий агент підключений до мережі моніторингу. Рух трафіку по звичайній мережі відображається в мережі моніторингу з використанням технології Switched Port Analyzer (SPAN). У проксі-сервері інтерфейси мережевого агента знаходяться у двох віртуальних машинах і трафік проходить безпосередньо через мережевий агент. Мережевий агент розгорнуто в Dom0 та вимкнено функцію дублювання трафіку в дзеркальному відображені.

Якщо Мережевий агент працює в Dom0 це є більш ефективним, так як він може прослуховувати трафік безпосередньо на віртуальному мосту. Проте, в DomU, трафік повинен бути продубльований на віртуальному інтерфейсі віртуальних машин (vif). Коли система виявлення вторгнень працює, мережевий агент споживає велику кількість системних ресурсів, так як він повинен перехоплювати весь трафік і виконувати перевірку пакетів. Для демонстрації оцінки продуктивності було проведено моніторинг таких показників: використання центрального процесора, затримки зв'язку в різних мережевих інтерфейсах.

Для імітації реального трафіку в системі хмари було використано генератор пакетів. Як показано на рис. 3.9, навантаження трафіку у вигляді швидкості передачі пакетів зростає від 1 до 3000 повідомлень за секунду. Аналіз графіку показує, що варіант реалізації на Dom0 споживає менше ресурсів центрального процесора, а режим роботи через DomU споживає максимальну кількість ресурсів процесора. Коли швидкість передачі пакетів досягає 3000 пакетів в секунду, використання ЦП при DomU досягає обмеження в 100%, в той час як режим DomU займає тільки близько 68% процесорного часу.

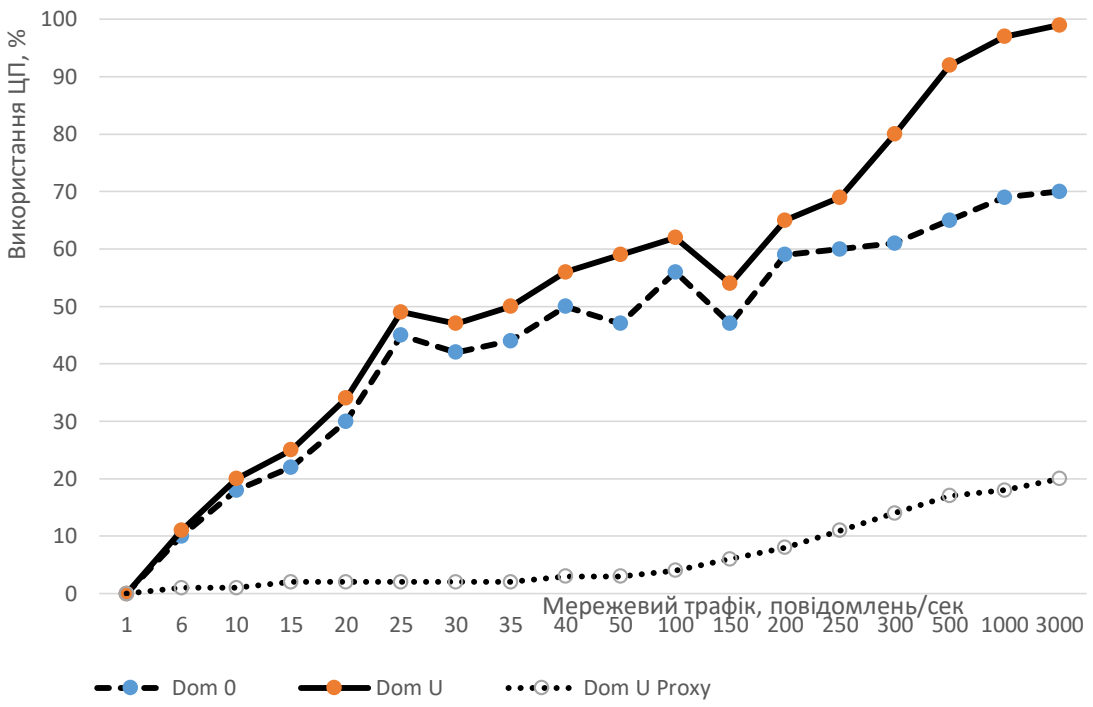

Рис. 3.9 - Використання центрального процесора в залежності від мережевого трафіку

На рис. 3.9, показана продуктивність роботи методики у відсотковому співвідношенні використання центрального процесору від успішно проаналізованих пакетів для випадків використання першого методу дзеркалювання трафіку в Dom0, DomU та другого методу використання проксі-

сервера. Таким чином, проведене дослідження показало, що використання першого методу з дзеркалюванням трафіку дасть змогу додатково використовувати 32% процесорного часу для виконання додаткових завдань, а застосування другого методу з проксі-сервером дає 80% вільного процесорного часу. Проте, у другому випадку це призведе до втрати часу на затримці пакетів при їх обробці.

На рис. 3.10 показана затримка зв'язку з системою при різних типах реалізації моніторингу мережевого середовища. Було створено 100 послідовних нормальних пакетів зі швидкістю 1 пакет в секунду, щоб перевірити затримку від початку до кінця обробки пакету на двох віртуальних машин в режимах віддзеркалення і проксі-сервера при роботі мережевого агенту методики забезпечення захисту віртуального хмарного середовища. На рис. 3.10 відображено мінімальну, середню і максимальну затримку зв'язку в порівняльному вигляді.

Результати дослідження показують, що затримка при проксі-сервері є найвища, тому що кожен пакет повинен пройти через нього безпосередньо. Варіант на основі відзеркалення трафіку на DomU та Dom0 не мають помітних відмінностей в затримці. Таким чином, мережевий агент на Dom0 та на DomU мають більш високу продуктивність з точки зору затримки.

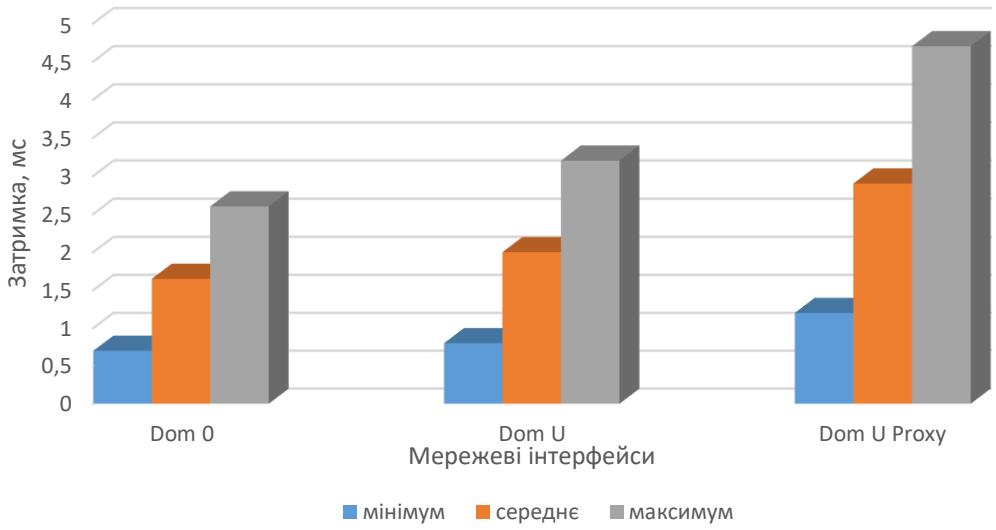

Рис. 3.10. Залежність затримки звязку у різних мережевих інтерфейсах

Експерименти доводять, що для приватної хмарної системи робота запропонованої методики не приводять до перенавантаження системи, що в свою чергу дає змогу виконання додаткових завдань.

Проведена оцінка продуктивності методики складається з двох частин. До першої частини відносяться показники оцінки захисту віртуального хмарного середовища. До другої частини відносяться порівняльна характеристика використання ЦП, оцінка пропускної здатності мережі, затримки при різних підходах реалізації паравіртуалізації. Отриманні результати дають можливість застосовувати методику для різних вимог щодо використання ЦП та величини затримки пакетів в системі.

Таким чином, доведено, що запропанована методика дозволяє досягти мети, що поставленна в роботі, а саме підвищити захист віртуальних хмарних ресурсів на основі моніторингу їх стану, оцінки напрямків протидії вторгнень та їх впливу на захист хмарного середовища.

Практичне значення проведенної роботи полягає у можливості використання розробленної методики для установки апаратних та програмних засобів в вірутальному хмарному середовищі, що дозволяє використовувати методику в центрах обробки даних. В майбутніх дослідженнях планується розробка децентралізованого підходу щодо забезпечення захисту віртуального хмарного середовища.

Висновки до розділу 3

1. У третьому розділі проведено розробку методики забезпечення захисту віртуального хмарного середовища, яка фокусується на збільшенні стійкості до компроментації віртуальних машин на основі виявлення атак та реконфігурації віртуальних мереж. Розроблена методика виявляє і запобігає вторгненям в розподілену віртуальну мережу в віртуальному хмарному середовищі. Вона дає можливість захопити і інспектувати підозрілий хмарний трафік без переривання призначених для користувача додатків і хмарних сервісів.

2. Методика забезпечення захисту віртуального хмарного середовища включає в себе рішення перемикання програмного забезпечення на карантин і перевірку підозрілих віртуальних машин для подальшого виявлення вторгнення та захисту. Через програмовану мережу методика підвищує ймовірність виявлення атаки і стійкість ВМ до експлуатаційних атак, не перериваючи сервісів хмари.

3. Методика граф атак для виявлення атак і профілактики використання вразливостей шляхом співвіднесення поведінки атаки, а також пропонує ефективні заходи протидії вторгненню та оптимізує реалізацію на хмарних серверах, щоб звести до мінімуму споживання ресурсів. Використання даної методики дає модливість використовувати неведику кількість обчислювальних потужностей.

4. Оцінка продуктивності системи доводить можливість методики знизити кількість компроментацій в віртуальній хмарній системи від зловживань з боку внутрішніх і зовнішніх атак.

5. Проведена практична реалізація розробленої методики. Для підтвердження адекватності отриманих результатів було розроблено тестове віртуальне хмарне середовище, на якому було перевірено розроблену методику. За результатами розрахунків середній показник зниження вразливості віртуальної машини за показником $F_{вм}$ складає 8 %, що дає можливість підвищити ефективність віртуального хмарного середовища.

ВИСНОВКИ

1. Досліджено хмарні віртуальні ресурси. Показано, що їх захист досягається на основі системного підходу до вирішення проблеми підвищення надійностних властивостей складних об'єктів управління.

2. Розроблено математичні моделі для забезпечення захисту віртуальних хмарних ресурсів, а саме: математичну модель впливу атаки на віртуальні хмарні ресурси, математичну модель оцінки стану віртуальних хмарних ресурсів, математичну модель вибору контрміри на основі комплексного показника для програмно-конфігурованих мереж. Для цього розроблено граф атак на віртуальне хмарне середовище. Цей граф дозволяє отримати інформацію про всі відомі вразливості системи, а також показує в режимі реального часу стан захисту системи, де ми можемо спрогнозувати можливі загрози і атаки шляхом кориляції виявлених подій.

3. Побудовано алгоритми: аналізу загроз у хмарному середовищі, результатом роботи якого є отримання одного або декількох шляхів проходження атаки на віртуальне середовище; вибору контрміри, який визначає вибір контрзаходів для конкретного сценарію атаки.

4. Вперше розроблено методику забезпечення захисту віртуального хмарного середовища, яка направлена на підвищення захисту до компрометації віртуальних машин на основі виявлення атак та реконфігурації віртуальних мереж.

5. Проведена практична реалізація розробленої методики. Для підтвердження адекватності отриманих результатів було розроблено тестове віртуальне хмарне середовище, на якому було перевірено розроблену методику. За результатами розрахунків середній показник зниження вразливості віртуальної машини за показником $F_{вм}$ складає 8 %, що дає можливість підвищити ефективність віртуального хмарного середовища.

Перелік посилань

1. «Big data: A review»,
[https://ieeexplore.ieee.org/document/6567202].
2. «Big Data & Society», Rob Kitchin, Gavin McArdle,
[https://journals.sagepub.com/doi/10.1177/2053951716631130].
3. «MapReduce: Simplified Data Processing on Large Clusters»,
[https://research.google/pubs/pub62/].
4. «Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing»,
[https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf].
5. «Spark SQL: Relational Data Processing in Spark»,
[https://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf].
6. «Apache Spark Website», [https://spark.apache.org/].
7. «Amazon Elastic Kubernetes Service (EKS)», [https://aws.amazon.com/eks/].
8. «Kubernetes», [https://kubernetes.io/].
9. «JSON Lines», [https://jsonlines.org/].
10. «Apache Parquet», [https://parquet.apache.org/].
11. «Spark Operator», [https://googlecloudplatform.github.io/spark-on-k8s-operator]
12. «Apache ZooKeeper», [https://zookeeper.apache.org/]
13. «Apache Storm», [https://zookeeper.apache.org/]
14. «Apache Flink», [https://flink.apache.org/]
15. «Apache Hive», [https://hive.apache.org]

ДОДАТОК А

/k8s/app.yaml

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: etl
  namespace: default
spec:
  type: Scala
  mode: cluster
  image: "267473891667.dkr.ecr.us-east-
1.amazonaws.com/spark/spark:latest"
  imagePullPolicy: Always
  mainClass: example.Main
  mainApplicationFile: "local:///opt/spark/jobs/etl.jar"
  dynamicAllocation:
    enabled: true
    initialExecutors: 2
    minExecutors: 2
    maxExecutors: 16
  sparkVersion: "3.1.1"
  timeToLiveSeconds: 43200
  nodeSelector:
    intent: spark
  hadoopConf:
    fs.s3a.impl: org.apache.hadoop.fs.s3a.S3AFileSystem
    fs.s3a.aws.credentials.provider:
"com.amazonaws.auth.WebIdentityTokenCredentialsProvider"
  restartPolicy:
    type: Never
  driver:
    cores: 1
    memory: 2048m
    # See: https://github.com/kubernetes/kubernetes/issues/82573
```

```
    # Note: securityContext has changed in recent versions of the
operator to podSecurityContext
    podSecurityContext:
      fsGroup: 65534
    securityContext:
      fsGroup: 65534
    labels:
      version: 3.1.1
    serviceAccount: spark
    annotations:
      fluentbit.io/exclude: "true"
  executor:
    instances: 1
    coreRequest: "500m"
    annotations:
      fluentbit.io/exclude: "true"
    memory: "1024m"
    # See: https://github.com/kubernetes/kubernetes/issues/82573
    # Note: securityContext has changed in recent versions of the
operator to podSecurityContext
    podSecurityContext:
      fsGroup: 65534
    labels:
      version: 3.1.1
```

/k8s/sa.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: spark
  namespace: default
  annotations:
    eks.amazonaws.com/role-arn:
arn:aws:iam::267473891667:role/universe-etl-manager
---
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: spark-jobs-role
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["*"]
  - apiGroups: [""]
    resources: ["services"]
    verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-jobs-role-binding
  namespace: default
subjects:
  - kind: ServiceAccount
    name: spark
    namespace: default
roleRef:
  kind: Role
  name: spark-jobs-role
  apiGroup: rbac.authorization.k8s.io
```

.vscode/settings.json

```
{
  "files.watcherExclude": {
    "**/target": true
  },
  "editor.tabSize": 2,
  "editor.insertSpaces": true
}
```

.vscode/launch.json

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "scala",
      "request": "launch",
      "name": "Run App",
      "mainClass": "spark.etl.Main",
      "args": [],
      "jvmOptions": [],
      "env": {}
    }
  ],
  "compounds": []
}
```

project/Dependencies.scala

```
import sbt._

object Dependencies {
  lazy val SparkVersion  = "3.2.0"
  lazy val HadoopVersion = "3.3.0"
  lazy val AwsSdkVersion = "1.11.797"

  lazy val spark = Seq(
    "org.apache.spark" %% "spark-core"      % SparkVersion  %
"provided",
    "org.apache.spark" %% "spark-sql"       % SparkVersion  %
"provided",
    "org.apache.spark" %% "spark-streaming" % SparkVersion  %
"provided",
```

```
    "com.amazonaws"     % "aws-java-sdk"    % AwsSdkVersion %
"provided",
    "org.apache.hadoop" % "hadoop-aws"      % HadoopVersion %
"provided",
    "org.apache.hadoop" % "hadoop-common"   % HadoopVersion %
"provided",
  )
}
```

src/main/scala/spark/Constants.scala

```
package spark.etl

object Constants {

  val SRC_BUCKET_NAME = "amplitude-export"

  val DST_BUCKET_NAME = "analitycs-storage"

}
```

src/main/scala/spark/Etl.scala

```
package spark.etl

import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SparkSession

import scala.util.Try
import scala.util.Failure
import scala.util.Success

trait ETLApp {

  def extract(spark: SparkSession): Try[DataFrame]
```

```
  def transform(df: DataFrame): Try[DataFrame]

  def load(df: DataFrame): Try[Unit]

  lazy val spark: SparkSession = SparkSession
    .builder()
    .getOrCreate()

  lazy val appName             = spark.sparkContext.appName

  def main(args: Array[String]): Unit = {
    println(s"ETL App $appName started")

    extract(spark).flatMap(transform).flatMap(load) match {
      case Failure(exception) => {
        println(s"ETL App $appName exits with an error")
        spark.close()
      }
      case Success(value)     => {
        println(s"ETL App $appName completed")
        spark.close()
      }
    }
  }
}
```

src/main/scala/spark/EventSchema.scala/:

```
package spark.etl

final case class AmplitudeEvent(
    $insert_id: String,
    $schema: Long,
    adid: String,
    amplitude_attribution_ids: Seq[String],
    amplitude_event_type: String,
```

```
amplitude_id: Long,
app: Long,
city: String,
client_event_time: String,
client_upload_time: String,
country: String,
data: Data,
device_brand: String,
device_carrier: String,
device_family: String,
device_id: String,
device_manufacturer: String,
device_model: String,
device_type: String,
dma: String,
event_id: Long,
event_properties: EventProperties,
event_time: String,
event_type: String,
idfa: String,
ip_address: String,
is_attribution_event: Boolean,
language: String,
library: String,
location_lat: Double,
location_lng: Double,
os_name: String,
os_version: String,
paying: String,
platform: String,
processed_time: String,
region: String,
sample_rate: String,
server_received_time: String,
server_upload_time: String,
```

```
    session_id: Long,
    start_version: String,
    user_creation_time: String,
    user_id: String,
    user_properties: UserProperties,
    uuid: String,
    version_name: String,
)

final case class Data(
    first_event: Boolean,
)

final case class EventProperties(
    ad_event_id: String,
    ad_id: String,
    adgroup: String,
    adgroup_id: String,
    adset: String,
    adset_id: String,
    advertising_id: String,
    af_ad: String,
    af_ad_id: String,
    af_ad_type: String,
    af_adset: String,
    af_adset_id: String,
    af_c_id: String,
    af_channel: String,
    af_click_lookback: String,
    af_ip: String,
    af_keywords: String,
    af_lang: String,
    af_message: String,
    af_mp: String,
    af_r: String,
```

```
af_reengagement_window: String,
af_siteid: String,
af_status: String,
af_ua: String,
af_viewthrough_lookback: String,
agency: String,
amlitude_only: Boolean,
answer: String,
as: String,
campaign: String,
campaign_id: String,
`click-timestamp`: String,
click_time: String,
clickid: String,
color: String,
comment: String,
cost_cents_USD: String,
counter: Long,
document_id: String,
error: String,
error_message: String,
extension: String,
external_account_id: Long,
finish_ts: String,
first_name: String,
format: String,
how: String,
http_referrer: String,
i_agree_promo_notifications_checkbox: String,
idfa: String,
install_time: String,
isPremium: String,
isTrial: Boolean,
is_completed: String,
is_fb: Boolean,
```

```
is_first_launch: Boolean,
is_mobile_data_terms_signed: Boolean,
is_paid: Boolean,
iscache: Boolean,
item: String,
lat: String,
link: String,
locked: String,
match_type: String,
media_source: String,
mode: String,
network: String,
orig_cost: String,
os: String,
page_count: String,
page_id: String,
pages_count: Long,
param: String,
product_id: String,
rate: Long,
redirect: String,
retargeting_conversion_type: String,
review: String,
sale_type: String,
screen_config_name: String,
screen_name: String,
second_name: String,
session_number: Long,
site: String,
status: String,
tier_name: String,
timestamp: String,
ts: String,
`type`: String,
userId: String,
```

```
    user_id: String,
    value: Double,
    via: String,
    watermark_mode: String,
    crop_count: Double,
    quad_diff: Double,
)

final case class UserProperties(
    ab_test: String,
    ad_campaign: String,
    ads_shown: String,
    def_link_screen_container: String,
    docs_counter: String,
    domain: String,
    export_counter: String,
    folders_counter: String,
    import_counter: String,
    install_date: String,
    is_premium: String,
    capi_test_name: String,
    capi_test_variant: String,
    late_purch_screen_loading_2_9_13: String,
    lifetime_vs_year_onb_screen_2_9_11: String,
    light_onboarding_flow_2_12_2: String,
    no_push_vs_push_onboarding_2_9_11: String,
    ocr_counter: String,
    ocr_saved: String,
    old_basic_screen_vs_apple_rec_screen_2_12_1: String,
    or_cont_with_lim_version_vs_x_2_9_14: String,
    pages_counter: String,
    push_status: String,
    regular_purch_flow_vs_fax_bundle_2_10: String,
    scaaan_purch_flow_2_9_13: String,
    scan_counter: String,
```

```
    security_type: String,
    session: String,
    sign_in_onboarding_flow_2_12: String,
    signatures_counter: String,
    switcher_purchase_screen_2_12_2: String,
    trial_14days_after_sale_conf_canceled_2_9_13: String,
    trial_7days_vs_14days_2_9_13: String,
    trial_7days_vs_3days_2_9_14: String,
    trial_vs_no_trial_screen_2_9_11: String,
    user_id: String,
    vertical_6_99_vs_vertical_9_99_version_2_9_2: String,
    vertical_vs_one_subscription_6_99_screen_2_9_9: String,
    vertical_vs_tile_screen_version_2_9: String,
    vertical_vs_winter_sale_screen_version_2_9_1: String,
    is_fb_capi: String,
    utm_campaign: String,
    site: String,
)
```

src/main/scala/spark/Main.scala

```
package spark.etl

import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SparkSession

import scala.util.Try
import org.apache.spark.sql.streaming.OutputMode
import org.apache.spark.sql.Encoders

object Main extends ETLApp {

  override def extract(spark: SparkSession): Try[DataFrame] = Try {
    val schema = Encoders.product[AmplitudeEvent].schema

    spark.readStream
```

```
      .schema(schema)
      .json(s"s3a://${Constants.SRC_BUCKET_NAME}")
  }

  override def transform(df: DataFrame): Try[DataFrame]     = Try {
    DFUtils.flattenDataFrame(df)
  }

  override def load(df: DataFrame): Try[Unit]               = Try {
    val _ = df.writeStream
      .outputMode(OutputMode.Append())
      .start(s"s3a://${Constants.DST_BUCKET_NAME}")
      .awaitTermination()
  }

}
```

src/main/scala/Utils.scala

```
package spark.etl

import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.types.StructType
import org.apache.spark.sql.Column
import org.apache.spark.sql.functions.col
import scala.annotation.tailrec
import org.apache.spark.sql.types.DataType

object DFUtils {

  def flattenDataFrame(df: DataFrame) = {
    def flattenSchema(
        schema: StructType,
        prefix: String = null
    ): Array[Column] = {
      schema.fields.flatMap(f => {
```

```
        val colName = if (prefix == null) f.name else (prefix + "."
+ f.name)

        f.dataType match {
          case st: StructType => flattenSchema(st, colName)
          case _              =>
Array(col(colName).as(colName.toString().replace(".", "_")))
        }
      })
    }

    val flattenedSchema = flattenSchema(df.schema)

    df.select(flattenedSchema: _*)
  }

}
```

.gitignore

```
/.bsp/
target/
```

.scalafmt

```
version = "3.0.5"

align.preset=most

align.openParenDefnSite = true
align.preset = more
align.multiline = true
trailingCommas = always

project {
  excludeFilters = [
```

```
    ".metals"
  ]
}

rewrite {
     rules = [
          SortModifiers
          sortimports
     ]
}

spaces {
     inImportCurlyBraces = true
}
```

build.sbt

```
import Dependencies._

ThisBuild / scalaVersion     := "2.13.7"
ThisBuild / version          := "0.1.0-SNAPSHOT"
ThisBuild / organization     := "com.example"
ThisBuild / organizationName := "example"

lazy val root = (project in file("."))
  .settings(
    name := "spark-etl",
    libraryDependencies ++= spark
  )

// See https://www.scala-sbt.org/1.x/docs/Using-Sonatype.html for
instructions on how to publish to Sonatype.
```