ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ ТА МЕРЕЖ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «МЕТОДИ ЗАХИСТУ ВІРТУАЛЬНИХ МЕРЕЖ ВІД АТАК»

на здобуття освітнього ступеня магістра
зі спеціальності 172 Електронні комунікації та радіотехніка
(код, найменування спеціальності)
освітньо-професійної програми Телекомунікаційні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олег ЛЕВЧИК
(підпис) *(Ім'я ПРІЗВИЩЕ)*

Виконав: здобувач вищої освіти гр. ТСДМ-62
Олег ЛЕВЧИК
Ім'я, ПРІЗВИЩЕ

Керівник: Вікторія КОРЕЦЬКА
к.п.н., доцент *Ім'я, ПРІЗВИЩЕ*

Рецензент: _____
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Телекомунікацій

Кафедра ___Телекомунікаційних систем та мереж_____
Ступінь вищої освіти __магістр_____
Спеціальність ___172 Електронні комунікації та радіотехніка _____
Освітньо-професійна програма _____Телекомунікаційні системи та мережі_____

ЗАТВЕРДЖУЮ
Завідувач кафедри __ТСМ_____
_____ Віктор ЗАЇКА
«____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Левчику Олегу Ігоровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Методи захисту віртуальних мереж від атак
керівник кваліфікаційної роботи Вікторія КОРЕЦЬКА, к.п.н., доцент
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» 10. 2025 р. № 467

2. Строк подання кваліфікаційної роботи «___» _____ 2025 р.

3. Вихідні дані до кваліфікаційної роботи:
   1. Захист віртуальних мереж.
   2. Машинне навчання.
   3. DDOS атаки.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Машинне навчання для захисту віртуальних мереж.
   2. Методи захисту відкритої інфраструктури.
   3. Захист віртуальної мережі L2 комутаторами.
   4. Захист віртуальних мереж від DDOS атак.

5. Перелік ілюстративного матеріалу: ___презентація на ____слайдах_____

6. Дата видачі завдання «___» _____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	15.10.2025	Викон.
2	Розробка першого розділу	20.10.2025	Викон.
3	Розробка другого розділу	30.10.2025	Викон.
4	Розробка третього розділу	10.11.2025	Викон.
5	Розробка четвертого розділу	20.11.2025	Викон.
6	Висновки, вступ, реферат	30.11.2025	Викон.
7	Розробка презентації	10.12.2025	Викон.

Здобувач вищої освіти _____ Олег ЛЕВЧИК
(підпис) *(Ім'я ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Вікторія КОРЕЦЬКА
(підпис) *(Ім'я ПРІЗВИЩЕ)*

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут телекомунікацій
ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач Левчик О.І. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 172 Електронні комунікації та радіотехніка
(код, найменування спеціальності)
освітньо-професійної програми Телекомунікаційні системи та мережі
(назва)
на тему: «Методи захисту віртуальних мереж від атак»

Кваліфікаційна робота і рецензія додаються.

Директор ННІТ _____ Владислав КРАВЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач Левчик Олег Ігорович показав гарну теоретичну та інженерну підготовку, уміння володіти новими комп'ютерними технологіями, користуватися навчальною, довідковою і науково-технічною літературою. Працюючи над завданнями, які були поставлені керівником, проявив ініціативність, сумлінність та хист до наукової роботи.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Левчика Олега Ігоровича на оцінку «відмінно» та присвоїти йому кваліфікацію магістр з електронних комунікацій та радіотехніки.

Керівник кваліфікаційної роботи _____ Вікторія КОРЕЦЬКА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«____» _____ 20___ року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Левчик О. І. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедри телекомунікаційних систем та мереж _____ Віктор ЗАЇКА
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

«____» _____ 20___ року

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну роботу
на здобуття освітнього ступеня магістра

здобувача вищої освіти Левчика Олега Ігоровича
(прізвище, ім'я, по батькові)
на тему «Методи захисту віртуальних мереж від атак»

Актуальність.
Тема роботи є актуальною, тому що стрімкий розвиток інформаційних технологій, хмарних обчислень і сервісів віртуалізації призвів до того, що віртуальні мережі стали невід'ємною складовою сучасних телекомунікаційних систем. Вони забезпечують масштабованість, гнучкість і оптимальне використання ресурсів, дозволяючи реалізовувати комплексні корпоративні й інфраструктурні рішення. Проте поряд із перевагами віртуальні середовища породжують нові виклики у сфері кібербезпеки, оскільки традиційні механізми захисту не завжди здатні ефективно протидіяти специфічним загрозам у таких умовах.
Позитивні сторони.
1. У роботі виконано всебічне дослідження сучасних загроз для віртуальних мереж, детально розглянуто принципи дії атак різних рівнів OSI-моделі, а також механізми їх виявлення та нейтралізації
2. У роботі досліджено технічних аспектів функціонування віртуальних середовищ, мережевих протоколів і апаратних механізмів захисту (RISC-V PMP, TrustZone, Spectre/Meltdown).
Недоліки.
1. Не у повному обсязі проведено кількісне зіставлення ефективності різних механізмів захисту, таких як DPI, SIEM, IDS/IPS, або методів машинного навчання для виявлення атак.
2. Робота не містить аналізу витрат на реалізацію запропонованих засобів захисту у віртуальних інфраструктурах, що знижує практичну придатність рекомендацій для підприємств або провайдерів.

Відзначені зауваження не впливають на загальну позитивну оцінку магістерської кваліфікаційної роботи

Висновок: *кваліфікаційна робота на здобуття ступеня магістра заслуговує оцінку "відмінно", а здобувач Левчик Олег Ігорович заслуговує присвоєння кваліфікації: магістр з електронних комунікацій та радіотехніки*.

Рецензент
науковий ступінь, вчене звання _____ (підпис) _____ *(Ім'я ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 83 стор., 17 рис., 25 джерел.

Мета роботи – систематизувати і оцінити сучасні підходи до захисту віртуальних мереж, розробити практично застосовну багаторівневу архітектуру захисту та запропонувати методики виявлення і протидії як класичним мережевим атакам, так і новим викликам, з урахуванням апаратних можливостей сучасних процесорних архітектур.

Об’єкт дослідження – віртуалізовані мережеві середовища та інфраструктури (включаючи віртуальні машини, віртуальні комутатори/маршрутизатори, SDN та хмарні платформи).

Предмет дослідження – методи, алгоритми та технічні механізми виявлення, запобігання і мінімізації наслідків атак у віртуальних мережах (мережеві політики, DPI, SIEM, поведінковий аналіз, захист на апаратному рівні, заходи проти adversarial ML).

Короткий зміст роботи: В роботі дослідженню методи захисту віртуальних мереж від сучасних кіберзагроз. Проаналізовано основні вектори атак, характерні для віртуалізованих середовищ, зокрема DDoS, ARP-spoofing, VLAN-hopping і підміну DHCP-серверів. Розглянуто вплив adversarial machine learning на ефективність систем виявлення атак. Досліджено апаратні засоби безпеки процесорів (RISC-V PMP, TrustZone, Spectre, Meltdown) та їх роль у захисті віртуалізованих систем. Запропоновано комплекс багаторівневих заходів — DHCP snooping, DAI, IP Source Guard, DPI, SIEM — для підвищення стійкості мереж. Результати підтверджують ефективність поєднання класичних і інтелектуальних методів захисту віртуальних мереж.

КЛЮЧОВІ СЛОВА: ВІРТУАЛЬНА МЕРЕЖА, МЕРЕЖЕВІ АТАКИ, ЗАХИСТ ДАНИХ, DDOS-АТАКА, MAC-SPOOFING, DHCP SNOOPING, DYNAMIC ARP INSPECTION, IP SOURCE GUARD, PORT-SECURITY, IDS/IPS, SIEM, DPI, МАШИННЕ НАВЧАННЯ, ADVERSARIAL MACHINE LEARNING

ABSTRACT

Text part of the master's qualification work: 83 pages, 17 figures, 25 sources.

The purpose of the work is to systematize and evaluate modern approaches to protecting virtual networks, develop a practically applicable multi-level protection architecture and propose methods for detecting and countering both classic network attacks and new challenges, taking into account the hardware capabilities of modern processor architectures.

The object of the study is virtualized network environments and infrastructures (including virtual machines, virtual switches/routers, SDN and cloud platforms).

The subject of the study is methods, algorithms and technical mechanisms for detecting, preventing and minimizing the consequences of attacks in virtual networks (network policies, DPI, SIEM, behavioral analysis, hardware-level protection, measures against adversarial ML).

Summary of the work: The work studies methods for protecting virtual networks from modern cyberthreats. The main attack vectors characteristic of virtualized environments are analyzed, in particular DDoS, ARP-spoofing, VLAN-hopping and DHCP server spoofing. The impact of adversarial machine learning on the effectiveness of attack detection systems is considered. Hardware security features of processors (RISC-V PMP, TrustZone, Spectre, Meltdown) and their role in protecting virtualized systems are studied. A set of multi-level measures is proposed — DHCP snooping, DAI, IP Source Guard, DPI, SIEM — to increase the resilience of networks. The results confirm the effectiveness of combining classical and intelligent methods of protecting virtual networks.

KEYWORDS: VIRTUAL NETWORK, NETWORK ATTACKS, DATA PROTECTION, DDOS-ATTACK, MAC-SPOOFING, DHCP SNOOPING, DYNAMIC ARP INSPECTION, IP SOURCE GUARD, PORT-SECURITY, IDS/IPS, SIEM, DPI, MACHINE LEARNING, ADVERSARIAL MACHINE LEARNING

ЗМІСТ

ВСТУП 9
РОЗДІЛ 1 МАШИННЕ НАВЧАННЯ ДЛЯ ЗАХИСТУ ВІРТУАЛЬНИХ МЕРЕЖ 11
1.1 Основні методи 11
1.2 Використання в віртуальних мережах 16
1.3 Категоризація загроз 27
РОЗДІЛ 2 МЕТОДИ ЗАХИСТУ ВІДКРИТОЇ ІНФРАСТРУКТУРИ 35
2.1 Еволюція архітектур віртуальних мереж 35
2.2 Захист пам’яті в архітектурах 39
2.3 Контроль захисту в процесорах віртуальних мереж 47
РОЗДІЛ 3 ЗАХИСТ ВІРТУАЛЬНОЇ МЕРЕЖІ L2 КОМУТАТОРАМИ 55
3.1 Методи захисту 55
3.2 Атаки DHCP starvation 61
3.3 Атаки CAM-table overflow 68
РОЗДІЛ 4 ЗАХИСТ ВІРТУАЛЬНИХ МЕРЕЖ ВІД DDOS АТАК 73
4.1 Атаки TCP SYN Flood 73
4.2 Атаки за участі ботнету 76
4.3 Інші види атак 79
ВИСНОВКИ 82
ПЕРЕЛІК ПОСИЛАНЬ 84
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 87

ВСТУП

Стрімкий розвиток інформаційних технологій, хмарних обчислень і сервісів віртуалізації призвів до того, що віртуальні мережі стали невід'ємною складовою сучасних телекомунікаційних систем. Вони забезпечують масштабованість, гнучкість і оптимальне використання ресурсів, дозволяючи реалізовувати комплексні корпоративні й інфраструктурні рішення. Проте поряд із перевагами віртуальні середовища породжують нові виклики у сфері кібербезпеки, оскільки традиційні механізми захисту не завжди здатні ефективно протидіяти специфічним загрозам у таких умовах.

Зловмисники активно намагаються експлуатувати особливості віртуалізованих інфраструктур для реалізації атак. До таких загроз належать DoS/DDoS-атаки, які призводять до перевантаження каналів та недоступності сервісів, а також більш специфічні техніки, як-от Smurf- та DNS-ампліфікація, MAC-spoofing, VLAN-hopping, CAM table overflow чи підміна DHCP-сервера. Попри різницю у реалізації, всі ці атаки можуть стати причиною серйозних порушень у роботі віртуальних систем і навіть зупинки критично важливих сервісів.

Окрему загрозу становлять маніпуляції над алгоритмами машинного навчання, які сьогодні активно інтегруються у системи безпеки для виявлення аномалій. Так зване adversarial machine learning може використовуватися для «переобучення» або введення моделей в оману, що значно знижує ефективність класичних систем виявлення атак. Це вимагає поєднання традиційних підходів (фільтрація, шифрування, контроль доступу) із сучасними методами поведінкової аналітики та глибокого аналізу трафіку.

Важливим напрямом у дослідженні є апаратні аспекти безпеки, адже вони безпосередньо впливають на стійкість віртуалізованих середовищ. У роботі розглянуто механізми захисту пам'яті, такі як PMP у RISC-V та Paging Protection у x86, що забезпечують контроль доступу до фізичних і віртуальних областей. Проаналізовано контроль привілеїв у різних архітектурах — від класичних рівнів

доступу у RISC-V і x86 до апаратної технології TrustZone у ARM, яка розділяє систему на захищене та звичайне середовище. Також досліджено методи протидії спекулятивним атакам типу Spectre та Meltdown, які вразили більшість сучасних процесорів і довели необхідність апаратних змін в архітектурі задля запобігання витоку конфіденційних даних. Ці механізми дозволяють локалізувати виконання потенційно небезпечних процесів і суттєво зменшити ризик несанкціонованого доступу до критичних ресурсів.

У цьому контексті питання забезпечення безпеки віртуальних мереж набуває особливої актуальності. Лише багаторівневий підхід, що поєднує класичні методи мережевої безпеки, інтелектуальні системи аналізу трафіку та апаратні механізми захисту сучасних процесорних архітектур, здатен забезпечити ефективну протидію як відомим, так і новим загрозам.

1 МАШИННЕ НАВЧАННЯ ДЛЯ ЗАХИСТУ ВІРТУАЛЬНИХ МЕРЕЖ

1.1 Основні методи

Одними з найпопулярніших і найобговорюваніших тем останніх років є — куди впровадили штучний інтелект і які системи вразили хакери. Поєднавши ці два напрями, у наукових дослідженнях з'являється багато цікавих робіт, що показують: моделі машинного навчання не є непомильними — їх можна вводити в оману або експлуатувати під час реальних атак. Наприклад, у наукових публікаціях детально розглядають випадки обмежень глибокого навчання та різні методи обману нейронних мереж, — від класичних атак на вхідні дані до більш витончених атак на навчальний процес.

Далі варто розглянути цю тему з погляду безпеки віртуальних мереж — не лише як окремий науковий інтерес, а як практичну загрозу для систем, що вже впровадили ML/AI у критичних або комерційних застосунках. Нижче наведено ключові питання, які допоможуть структурувати подальший огляд і дослідження:
Важливі терміни. — коротке пояснення основних понять (адверсарні приклади, отруєння даних, евазійні атаки, model extraction, membership inference тощо) дає спільну мову для фахівців із безпеки й розробників моделей.

- Що таке машинне навчання— стисле пояснення принципів: навчальна вибірка, функція втрат, процес оптимізації, узагальнення на нових даних; важливо розуміти, де саме виникають точки вразливості.
- А причому тут безпека віртуальних мереж — моделі ML/AI стають частиною ланцюга прийняття рішень, обробки персональних або технічних даних, і помилка або зловживання ними може призвести до фінансових втрат, порушення приватності або навіть до фізичних наслідків у кібер-фізичних системах.
- Чи можна маніпулювати моделлю машинного навчання, щоб провести цільову атаку? — так; у наукових роботах показано методи створення спеціально спотворених входів (адверсарних прикладів) і отруєння

тренувальних даних, які дозволяють змусити модель робити передбачувані помилки для конкретних цілей.

- Чи можна погіршити продуктивність системи? — окрім цілеспрямованих атак, існують атаки на доступність (denial-of-service-подібні сценарії для ML), а також масштабні маніпуляції даними, що знижують точність і узагальнення моделі в часі.
- Чи можна скористатися обмеженнями моделей машинного навчання? — моделі мають обмеженість у здатності інтерполювати поза розподілом навчальних даних; ці «сліпі зони» дослідники використовують для розробки атак і для виявлення ризиків експлуатації в реальних сценаріях.
- Категоризація атак. — корисно розділяти атаки за вектором (вхідні/тестові, отруєння тренувальних даних, компрометація моделі, витік даних), за ціллю (помилкова класифікація, вивід конфіденційної інформації, відмова в обслуговуванні) і за рівнем доступу атакуючого (білі-ящики, сірі-ящики, чорні-ящики).
- Способи захисту. — існують як прикладні підходи (фільтрація та аугментація даних, adversarial training, обмеження доступу до API, моніторинг поведінки моделі), так і формальні методи (сертифікована робастність, differential privacy, механізми контролю версій та аудиту моделей). Сукупність технічних, організаційних і процедурних заходів дає найкращий ефект.
- Можливі наслідки. — від погіршення якості сервісу й фінансових збитків до витоку персональних даних і шкоди для фізичних об’єктів у критичних інфраструктурах; тому питання захисту моделей як частини загальної безпекової стратегії стає критично важливим.

Почати варто з уточнення термінології, адже саме від коректного перекладу й трактування ключових понять залежить точність усієї подальшої дискусії.

Варто зауважити, що питання перекладу терміна «adversarial» уже неодноразово ставало предметом дискусій у науковому та професійному середовищі. Традиційно його перекладають юридичним терміном «змагальний», однак такий варіант є радше формальним і не розкриває сутності явища у сфері інформаційної безпеки.

Якщо ж застосувати підхід, орієнтований на кіберзахист, то більш адекватним буде переклад «шкідливий» або навіть «ворожий», оскільки саме ці визначення найкраще відображають характер атак, які мають на меті навмисно маніпулювати поведінкою системи.

У цьому ж контексті термін «adversarial intelligence» доцільно перекладати як «розвідка противника». Це поняття вже давно використовується в галузі військової та кібернетичної безпеки для опису діяльності, пов’язаної з аналізом можливостей, методів та стратегій супротивника. Якщо застосувати таке розуміння до штучного інтелекту, воно дає змогу побачити, що adversarial-підходи — це не просто «технічна гра» з даними, а продумана тактика впливу на роботу моделей.

Правильне тлумачення змінює і сприйняття супутніх термінів:

- **«adversarial example»** — не просто «приклад для змагання», а «шкідливий екземпляр даних», тобто спеціально модифікований набір інформації, покликаний ввести модель в оману;
- **«adversarial settings»** — не «змагальні умови», а «шкідливе середовище», яке навмисне створюється для перевірки чи підриву стабільності системи;
- **«adversarial machine learning»** — доцільно називати «шкідливим машинним навчанням» або навіть «ворожим машинним навчанням», що підкреслює його прикладну спрямованість у контексті атак і маніпуляцій.

Саме таке уточнення термінології дозволяє чіткіше побачити, що мова йде не лише про теоретичні дослідження у сфері штучного інтелекту, а й про прикладні проблеми інформаційної безпеки. Адже adversarial-техніки фактично перетворюють машинне навчання на арену протистояння між розробниками моделей і потенційними зловмисниками.

У межах даної роботи будуть використовуватися саме такі переклади українською мовою. Такий підхід виправданий тим, що він робить терміни більш прозорими й прикладними, дозволяє уникнути двозначностей і демонструє реальну небезпеку, яка криється за дослідженнями adversarial-методів. Адже ця тематика значно ближча до проблематики захисту інформації, ніж до формальної

юриспруденції чи абстрактної лінгвістики, і тому природно застосовувати саме терміни з галузі телекомунікацій, а не буквальні кальки з англійської.

Методи машинного навчання прийнято визначати як сукупність підходів до створення алгоритмів, здатних автоматично навчатися та приймати рішення без детального програмування кожного кроку. Вони ґрунтуються на аналізі попередньо зібраних даних і виявленні закономірностей у них. Під поняттям *дані* можна розуміти майже все: числові значення, текстові записи, зображення, відео, сигнали чи будь-які інші сутності, якщо їх можливо описати через певні характеристики або вимірювані параметри. Якщо ж існує якась характеристика, яка невідома для частини об’єктів, але має велике значення, тоді застосування методів машинного навчання дозволяє, використовуючи вже відому інформацію, відновити або спрогнозувати відсутній параметр. Це робить машинне навчання одним із найпотужніших інструментів для сучасної аналітики та автоматизації.

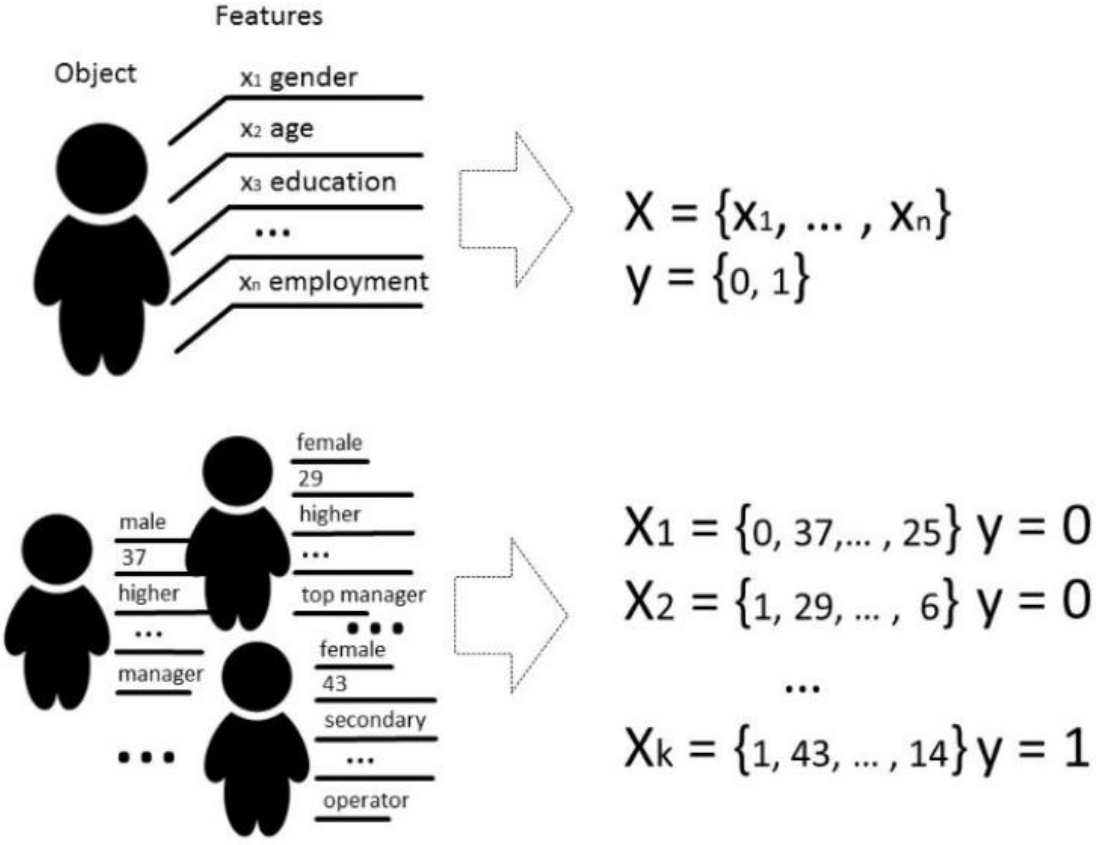


Рис. 1.1 Приклад машинного навчання для відновлення або прогнозування даних

Задачі, які вирішуються за допомогою машинного навчання, досить різноманітні. Серед них можна виділити класифікацію, регресію, кластеризацію, прогнозування часових рядів, а також навчання з підкріпленням. Однак у межах цього огляду основну увагу зосередимо саме на задачі класифікації, оскільки вона є однією з найпоширеніших і найбільш наочних для розуміння принципів роботи алгоритмів.

Метою етапу навчання моделі-класифікатора є пошук такої математичної залежності (функції), яка б встановлювала чіткий зв’язок між набором ознак конкретного об’єкта та його належністю до певного класу. У простішому випадку результат — це однозначне віднесення об’єкта до однієї з категорій. У складнішому сценарії алгоритм має передбачити ймовірність належності об’єкта до тієї чи іншої групи, що робить модель більш гнучкою й адаптивною.

Суть задачі класифікації можна пояснити через побудову гіперплощини — межі, яка розділяє простір ознак так, щоб елементи різних класів знаходилися по різні її сторони. Простір ознак має таку кількість вимірів, яка відповідає кількості параметрів (довжині вектора ознак), що описують об’єкт. У випадку двовимірного простору ця гіперплощина є простою прямою лінією, яка відділяє одну групу даних від іншої.

Розглянемо простий приклад.

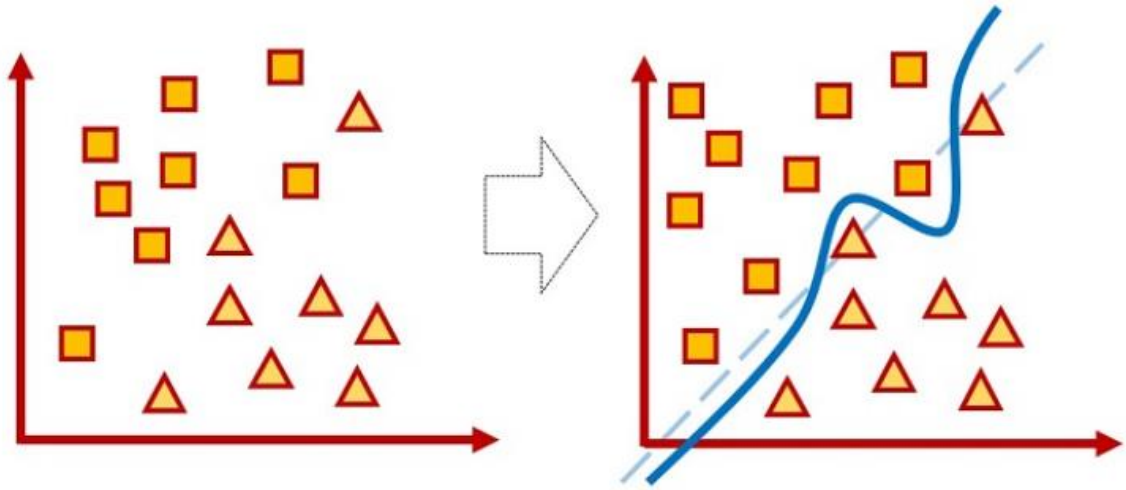


Рис. 1.2 Два класи об’єктів

Припустимо, що на ілюстрації наведено два класи — квадрати та трикутники. Застосування лінійної функції для поділу цих об’єктів виявляється недостатнім, оскільки дані накладаються одне на одне і не можуть бути розділені простою прямою. У цьому випадку методи машинного навчання дозволяють підібрати нелінійну функцію, яка здатна більш точно провести межу й розділити множини. Цей підхід забезпечує точніше прогнозування навіть у складних випадках, коли дані мають складну структуру.

1.2 Використання в віртуальних мережах

У сучасних віртуальних мережах методи машинного навчання відіграють дедалі важливішу роль, адже вони застосовуються для цілого спектра завдань — від інтелектуальної маршрутизації трафіку та балансування навантаження до виявлення аномалій, захисту від фроду чи шкідливого програмного коду. Завдяки їм стає можливим підвищення надійності та ефективності мережевої інфраструктури, що особливо актуально в умовах швидкого зростання обсягів переданої інформації та збільшення складності топологій.

Функціонування віртуальної мережі в цьому аспекті нагадує своєрідну гру, де кожен зроблений крок — це налаштування або зміна політики маршрутизації, — викликає реакцію «супротивника», яким можуть бути непередбачувані умови трафіку, кібератаки або внутрішні відмови. Тому доводиться постійно коригувати алгоритми керування, доповнювати їх новими даними та методами, а в окремих випадках — повністю перебудовувати моделі з урахуванням новітніх наукових підходів у сфері штучного інтелекту та телекомунікацій.

Якщо традиційні системи управління мережами ґрунтувалися на жорстко прописаних правилах, евристиках і сигнатурах, які вимагали постійної ручної підтримки, то сьогодні вони втрачають ефективність. Правила легко обходяться — наприклад, за допомогою складних методів шифрування, обфускації або модифікації пакетів даних. У результаті оператори віддають перевагу більш інтелектуальним рішенням, заснованим на машинному навчанні, які можуть автоматично виділяти приховані ознаки, швидко обробляти великі масиви інформації, знаходити закономірності та оперативно приймати рішення в реальному часі.

Таким чином, з одного боку, машинне навчання у віртуальних мережах використовується як потужний інструмент захисту та оптимізації, що підвищує стійкість і продуктивність системи. З іншого боку, ці ж самі методи можуть застосовуватися і для створення більш інтелектуальних атак, які враховують особливості топології, слабкі місця протоколів чи обмеження ресурсів. Це робить взаємодію машинного навчання та віртуальних мереж не статичним процесом, а динамічним змаганням, у якому результат визначається швидкістю адаптації, гнучкістю алгоритмів та здатністю своєчасно реагувати на нові виклики.

Спробуймо подивитися глибше: чи дійсно інструменти машинного навчання є вразливими, і які фактори впливають на їхню стійкість.

Для будь-якого алгоритму ключовим чинником є не лише сам процес налаштування параметрів чи вибір оптимальної архітектури, але й якість та структура даних, на основі яких відбувається навчання. У теоретично ідеальних умовах бази даних мають бути достатньо великими, класи — рівномірно

збалансованими, а навчання проходити без суттєвих витрат часу. Проте у реальному світі такі умови є радше винятком, ніж правилом: дані часто неповні, нерівномірно представлені, мають шум або приховані закономірності, які складно врахувати.

Якість навченої моделі зазвичай визначається через точність класифікації на тестових даних, яких система ще не бачила. У більш загальному вигляді — це співвідношення правильно класифікованих прикладів до загальної кількості даних, поданих моделі. Але ця метрика дає лише усереднену оцінку продуктивності й мало що говорить про поведінку системи у стресових чи нетипових сценаріях.

Проблема в тому, що стандартні оцінки якості завжди ґрунтуються на припущенні про очікуваний розподіл вхідних даних, тоді як у реальності він може суттєво змінюватися. Особливо це стосується так званих шкідливих середовищ (adversarial settings), де дані спеціально модифікуються для того, щоб система почала давати помилкові результати. Прикладами подібних середовищ є механізми протидії спам-фільтрам, обхід систем виявлення шахрайських операцій чи маніпуляції в алгоритмах ідентифікації шкідливого програмного забезпечення.

Точність у таких випадках можна трактувати як показник «середньої продуктивності» моделі під час звичайного використання. Але якщо говорити про безпеку алгоритму, то тут нас цікавить його поведінка у найгірших умовах — тоді, коли середовище навмисно налаштоване проти нього. І саме там найчастіше проявляються критичні вразливості.

Звичайне тестування моделей машинного навчання передбачає статичне середовище, де співвідношення між класами стабільне та відоме. Але на практиці ніхто не може гарантувати подібної стабільності. Більше того, для дослідників безпеки важливим завданням є якраз пошук таких вхідних даних, які змушують модель помилятися. Зловмисники свідомо намагаються створити якомога більше «підривних» векторів, здатних змусити систему дати неправильний прогноз.

Коли ми говоримо про захищеність інформаційних систем, зазвичай мається на увазі, що політика безпеки не може бути порушена в межах заданої моделі загроз на рівні апаратного чи програмного забезпечення. Традиційно системи тестують на етапі розробки й під час експлуатації. Проте сучасні сервіси все частіше спираються

на алгоритми обробки й аналізу даних. Це означає, що потенційні ризики приховані не лише у вразливостях самої функціональності, але й у даних, на основі яких модель приймає рішення. Саме дані стають новою ціллю атак.

Слід також враховувати, що зловмисники постійно вдосконалюють свої методи. Тому виникла окрема дослідницька галузь — adversarial machine learning, або ж *шкідливе машинне навчання*. Її предметом є методи аналізу та перевірки алгоритмів на здатність протистояти компрометації, коли атакувальник може використати знання про внутрішню логіку роботи моделі. Це не просто академічний інтерес, а критично важливий напрям для всіх сфер, де системи машинного навчання застосовуються в практичних цілях — від віртуальних мереж і фінансових технологій до медицини та транспортних систем.

Таким чином, безпека моделей машинного навчання виходить за межі класичної оцінки точності чи ефективності алгоритмів. Вона охоплює комплекс питань: наскільки стійкою буде система в умовах навмисного втручання, як реагуватиме на невідомі або штучно змінені дані, і чи здатна вона працювати надійно у випадку цілеспрямованої атаки. Саме ці аспекти й формують основу сучасних досліджень у сфері шкідливого машинного навчання.

Чи можна маніпулювати моделлю машинного навчання з метою цілеспрямованої атаки — і як це стосується віртуальних мереж?

Так — і найкраще це ілюструють кілька практичних сценаріїв. Почнемо з найпростішого прикладу, знайомого багатьом: пошукова оптимізація. Досвідчені оператори сайту вивчають, як працюють алгоритми ранжування, і підлаштовують контент під їхню логіку, щоб піднятися у видачі. Це навряд чи становить безпекову кризу саме по собі, поки не призводить до масштабної маніпуляції результатами пошуку або компрометації довіри до сервісу. Але концепція та сама — знаючи «вразливості» алгоритму, можна керувати його поведінкою.

Особливо небезпечні сценарії виникають у системах з онлайн-навчанням, де модель оновлює свої параметри в режимі реального часу на підставі послідовно надходячих даних. Зловмисник, який розуміє механіку цього процесу, може подавати підготовлені потоки даних, поступово зміщуючи розподіл і

«перенавчаючи» модель на потрібний результат. Такі атаки називають «отруєнням даних» (data poisoning) — і їхня сила полягає в тому, що зміни відбуваються непомітно, крок за кроком.

Приклади з біометричних систем добре демонструють цей механізм: багатьом системам властиве поступове оновлення шаблонів розпізнавання (щоб враховувати природні зміни зовнішності). Зловмисник може зловживати цією функцією, підставляючи модифіковані зразки так часто, щоб модель почала відносити його атрибути до профілю жертви — в результаті зловмисник отримує можливість імітувати жертву.

Те саме стосується і віртуальних мереж (SDN/NFV, хмарні сегменти, ізольовані віртуальні середовища). У таких системах алгоритми ML застосовують для класифікації трафіку, виявлення аномалій, політик маршрутизації, балансування навантаження та автоматичного налаштування ресурсів. Ось кілька реалістичних векторів атак у контексті віртуальних мереж:

- **Отруєння телеметрії та сигналів моніторингу.** Якщо модель, що виявляє аномалії, навчається або адаптується на основі мережевих метрик (флоути, затримки, показники помилок), зловмисник може фальсифікувати ці дані — змусивши систему «звикнути» до аномалій і перестати їх помічати, або навпаки — викликати хибні спрацьовування і відключення сервісів.
- **Маніпуляція сигналами контролера SDN.** Контролери SDN часто приймають рішення на основі статистики трафіку та моделей поведінки. Атака, яка підміщує або спотворює пакети телеметрії, може привести до помилкового перенаправлення трафіку, створення «чорних дір» або обходу політик безпеки.
- **Цілеспрямоване спотворення класифікаторів трафіку.** Багато систем QoS та DDoS-захисту покладаються на ML-класифікатори для відокремлення корисного та шкідливого трафіку. Поступове подавання спеціально сформованих зразків трафіку може змусити модель неправильно маркувати шкідливі потоки як легітимні — і навпаки.

- **Атаки на оркестрацію NFV.** Сервіси віртуальної мережі (VNFs) і їхня оркестрація залежать від прогнозів завантаження. Маніпулюючи вхідними метриками, можна викликати невиправдане масштабування або, навпаки, недомасштабування критичних функцій.
- **Компрометація політик сегментації.** Якщо ML-моделі допомагають автоматично сегментувати мережу під робочі навантаження, «перенавчання» цих моделей може призвести до некоректного об’єднання зон довіри і, як наслідок, розширення поверхні атаки.

Наслідки таких атак у віртуальних мережах можуть бути багаторівневими: від деградації продуктивності й помилкових сповіщень до витоку даних, обходу засобів захисту і навіть фізичних перебоїв у сервісах, що залежать від мережі.

Важливо підкреслити: головна особливість цих атак — тонка, поетапна модифікація поведінки системи. На відміну від класичної «вибухової» атаки, тут зміни накопичуються, часто проходячи під радаром стандартних тестів і метрик. Саме тому для захисту потрібні не лише оцінки середньої точності, а й аналіз стійкості до найгірших сценаріїв та підхід «security-by-design» при інтеграції ML у мережеві компоненти.

Як коротке резюме: так, можливо маніпулювати моделями задля цілеспрямованих атак — і у віртуальних мережах це загрожує як локальними помилками (помилкові маршрути, помилкова класифікація), так і масштабними наслідками (обхід захисту, витік чи втрата доступності сервісів). Якщо хочеш, я можу далі розширити цей розділ конкретними техніками атак (data poisoning, model extraction, evasion attacks) та запропонувати заходи захисту, специфічні для віртуальних мереж.

Чи може зловмисник підібрати такі «валідні» (з вигляду коректні) дані, що модель почне систематично помилятися й продуктивність системи впаде настільки, що її доведеться вимкнути?

Так — така загроза реальна, і її варто розглядати дуже серйозно. Нижче — розгорнутий, структурований опис, чому зловмисник може підібрати «валідні»

дані, які систематично знижують продуктивність моделі до критичного рівня, які існують вектори атак, можливі наслідки та практичні заходи захисту.

Чому це взагалі можливо

- **Оцінка якості прив'язана до розподілу даних.** Більшість моделей тестують і валідують на наборі даних, що імітує очікуваний розподіл. В реальному житті потоки даних змінюються (distribution shift), і модель, хороша на тренувальному наборі, може катастрофічно впасти на нових прикладах.
- **Модель працює з ймовірностями.** Алгоритм дає не абсолютну істину, а оцінку ймовірності належності до класу, тому завжди є місце для помилок першого (false positive) і другого (false negative) роду.
- **Наявність «гірших» наборів даних.** Існує множина вхідних векторів, на яких модель показує найгірший результат — і для атакуючого завдання полягає в пошуку або генерації таких векторів.
- **Онлайн-адаптація та телеметрія.** Якщо система навчається або адаптується в реальному часі (онлайн learning, continual learning) або приймає рішення на основі потокової телеметрії, малими кроками можна змістити поведінку моделі непомітно для операторів.
- **Людський фактор.** Висока кількість хибних спрацювань призводить до «втоми від оповіщень» — оператори знижують чутливість або взагалі відключають захист.

Типи атак і техніки маніпуляції

- **Отруєння даних (data poisoning).** Вставка в тренувальний або оновлювальний потік підготовлених прикладів, що змінюють кордони прийняття рішень моделі (може бути поступове «тонке» отруєння або «грубий» вкид).
- **Поступовий дрейф (targeted drift).** Серія коректних на вигляд, але спланованих подій, які змінюють статистику вхідних ознак і «приучають» модель до нових шаблонів.

- **Навмисне створення краєвих випадків.** Генерація прикладів, що лежать на межі розподілу ознак, де модель найменш упевнена — це підвищує частоту помилок.
- **Flooding / alert fatigue.** Масоване створення численних «близьких до підозрілих» подій, що провокує лавинні false positive та витрати на розслідування.
- **Evasion (омана класифікатора).** Модифікація валідних об’єктів (файлів, мережевих пакетів, зображень), щоб обійти детектори — приклад: адверсарні збурення для зображень або мінімальні зміни в заголовках мережевих пакетів.
- **Атаки на метадані і телеметрію.** Підміна або підробка вимірів, що подаються в систему моніторингу/контролера (наприклад, в SDN/NFV), що призводить до хибних оптимізацій або до втрати видимості реальної ситуації.

Реалістичні приклади (українська практика + віртуальні мережі)

- **Антивірусні рішення:** серія легітимних файлів із особливими властивостями викликає багато false positives → адміністратори знижують чутливість або відключають захист.
- **Фрод-системи в фінтехі:** атакуючий генерує транзакції, що статистично нагадують нормальні, але поступово змінюють профіль — модель перестає помічати справжній фрод.
- **Віртуальні мережі (SDN / NFV / хмара):** підроблена телеметрія або цілеспрямовані потоки трафіку призводять до неправильних рішень контролера (перенаправлення, масштабування VNFs, некоректна сегментація) — наслідок: деградація сервісів, обхід політик безпеки або зниження доступності.

Наслідки атак

- **Операційні:** зростання числа помилкових інцидентів, навантаження на аналітиків, затримки реакції, втрата довіри до системи.
- **Безпекові:** пропуск реальних атак, підвищений ризик витоку даних, обходи захисних механізмів.

- **Бізнес:** репутаційні збитки, фінансові втрати, юридична відповідальність (особливо в критичних галузях: медицина, банкінг, енергетика).

Практичні заходи захисту (технічні та організаційні)

1. **Робастне тестування (robustness & adversarial evaluation).** Стрес-тести на worst-case наборах, симуляція отруєнь і атак, adversarial validation.
2. **Adversarial / robust training.** Навчання з урахуванням адверсарних прикладів; використання методів, що підвищують стійкість до збурень.
3. **Розмаїтість валідаційних наборів.** Перевірка моделі на мультидоменних наборах даних та сценаріях зі зміною розподілу.
4. **Моніторинг дрейфу та метрик здоров'я моделі.** Постійний контроль distribution drift, confidence scores, calibration; оповіщення при неприродних змінах.
5. **Фільтрація та перевірка якості навчальних даних.** Автоматизовані детектори аномалій в потоці даних; валідація джерел; підписування / автентифікація телеметрії.
6. **Phased / canary deployment і human-in-the-loop.** Поступове розгортання моделей (canary), контроль оператором для критичних рішень, легкий механізм відкату.
7. **Ensemble та контроль консенсусу.** Використання ансамблів моделей або додаткових перевірочних механізмів, щоб зменшити вразливість однієї моделі.
8. **Лімітування та квотування подачі даних.** Обмеження швидкості і обсягу даних від одного джерела, щоб ускладнити масовані ін'єкції.
9. **Логи, аудит і forensics.** Краща видимість — докладні логи змін у моделі, вже застосованих даних, рішень і відкатів.
10. **Оновлення політик і навчання персоналу.** Процедури реагування, тренінги для аналітиків, врахування соціального інженерингу (human factor).

Рекомендації щодо архітектури і процесів

- Проєктуйте ML-сервіси з принципом **security-by-design**: припускайте, що дані можуть бути ворожими.

- Впровадьте **кілька рівнів захисту**: передова детекція (ML), перевірки на правилах, ручний аудит для критичних спрацьовувань.
- Встановіть **SLAs для безпеки** моделі: пороги деградації, автоматичні відключення/переключення на резервні алгоритми.
- Плануйте **регулярні чергові перевірки** (red teaming) — незалежні тести, що намагаються знайти worst-case входи.

Зловмисник цілком може знайти або згенерувати «валідні» дані, які поступово підточують модель до критичного стану, після чого систему вимикають. Це загрожує як операційній надійності, так і інформаційній безпеці. Захист вимагає поєднання технічних підходів (робастне навчання, моніторинг, фільтрація), архітектурних рішень (canary, ensemble, лімітування) та організаційних політик (аудит, інструкції, навчання персоналу).

Останні дослідження показали реальну можливість цілеспрямованих атак на системи автоматичного розпізнавання мовлення (ASR). Голосові повідомлення стали популярним форматом спілкування в соціальних мережах, але не завжди зручно їх прослуховувати — тому з’явилися сервіси, що автоматично трансформують аудіо в текст. Саме ці сервіси виявилися потенційною мішенню для атак: дослідники навчилися аналізувати оригінальні аудіозаписи та акустичні властивості сигналу, а потім дозовано додавати такі невеликі модифікації, що змінений сигнал залишається практично не відрізним для вуха, але приводить систему розпізнавання до бажаного результату.

Технічно це досягається шляхом підбору адверсарних збурень — тонких, часто непомітних змін у амплітуді або спектральних характеристиках сигналу, оптимізованих під конкретний ASR-класифікатор. У роботах згадується, що змінений сигнал може бути на ~99% схожим на оригінал з точки зору акустичних метрик або людського сприйняття, але в той же час модель транскрипції видає зовсім інший текст — іноді з формулюванням, яке вигідне або шкідливе для атакувальника.

Ці атаки бувають як цілеспрямованими (attacker specifies, який саме текст має отримати система), так і евейзійними (завдання — змусити модель не розпізнати

або змаркувати важливу інформацію). Важливий фактор — переносимість: збурення, підібрані для однієї моделі, іноді працюють і проти інших систем розпізнавання, особливо якщо вони використовують схожі акустичні ознаки або мережеві архітектури. Крім того, атаки можуть бути реалізовані в «реальному світі» — наприклад, шляхом відтворення модифікованого аудіо через динамік у присутності мікрофона сервісу, що робить загрозу практичною, а не лише теоретичною.

Наслідки таких атак — від неправильної транскрипції приватних розмов до маніпуляції голосовими командами (наприклад, для голосових помічників) або підміни доказів у судових розслідуваннях. Тому у відповідь на ці загрози пропонують кілька напрямів захисту: адверсарне навчання (додавання збурень у тренувальний процес), попередня обробка сигналу (фільтри, усереднення, перевірка консистентності), застосування моделей zrobust features, багаторівнева валідація транскрипції (перехресна перевірка різними системами) та детектори адверсарних прикладів. Також практикують тестування моделей у фізичних умовах і впровадження політик «human-in-the-loop» для критичних рішень.

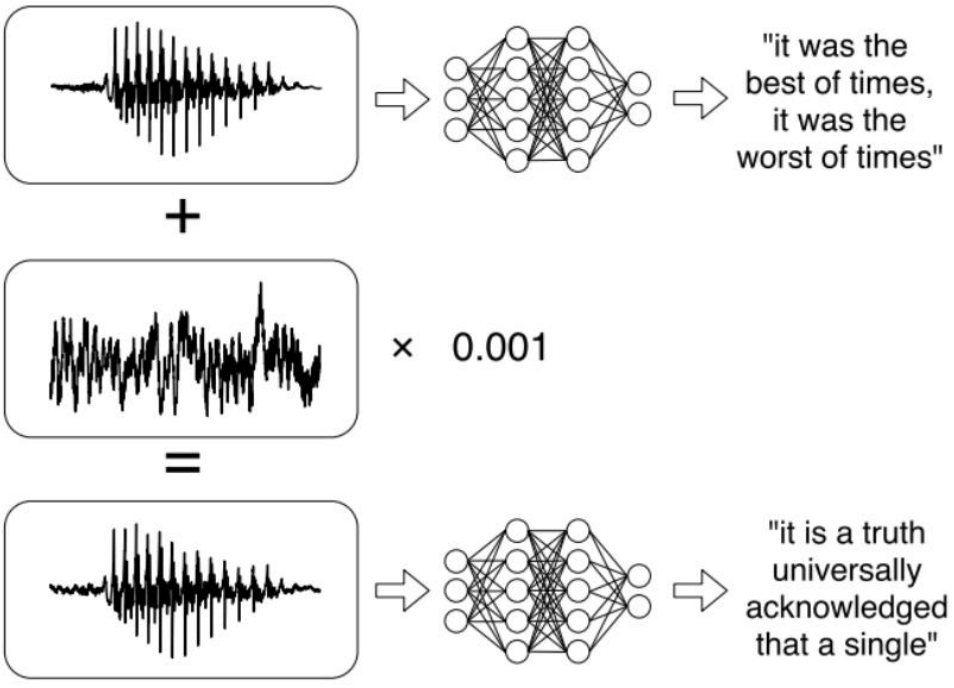


Рис. 1.3 Приклад цілеспрямованої атаки на систему автоматичного розпізнавання мовлення (ASR)

1.3 Категоризація загроз

Нижче — детальний виклад того, як можна класифікувати атаки на системи з ML, які цілі безпеки вони порушують, приклади застосування ML у захисті та практичні рекомендації щодо мінімізації ризиків.

1. За способом впливу (Influence)

- **Каузальні атаки (Causative attacks)** — втручання на етапі навчання: атакуючий додає або змінює зразки в тренувальній вибірці або в потоках для online-навчання, що призводить до зміни кордонів прийняття рішень моделі та довготривалої деградації її поведінки.
- **Дослідницькі атаки (Exploratory attacks)** — атакуючий не змінює дані для навчання, а натомість вивчає та експлуатує слабкі місця натренованої моделі: підбирає вхідні приклади, які викликають некоректні рішення, проводить probing або model extraction.

2. За типом порушення безпеки (Security violation)

- **Порушення цілісності (Integrity attacks)** — мета: змусити систему пропустити шкідливе (false negative). Це приводить до прямої компрометації ресурсів (наприклад, пропуск вірусного файлу або шахрайської транзакції).
- **Порушення доступності (Availability attacks)** — мета: викликати багато помилкових спрацьовувань (false positive), дезорганізувати роботу або змусити адміністраторів відключити захист (наприклад, flood-атаки, які призводять до «втоми від оповіщень»).
- **Порушення конфіденційності (Privacy attacks)** — витік або екстракція приватної інформації через модель (membership inference, model inversion), що також важливо враховувати як окрему категорію загроз.

3. За специфікою цілі

- **Таргетована атака (Targeted attack)** — атакуючий прагне досягти конкретного неправильного передбачення (наприклад, щоб біометрична система ототожнила зловмисника з конкретною жертвою).

- **Масова/безрозбірлива атака (Indiscriminate attack)** — мета — зламати роботу моделі в широкому масштабі, щоб багато легітимних прикладів класифікувалися неправильно.

4. Приклади застосування ML у безпеці

- **Антивірусні системи** — класифікація файлів/процесів як «шкідливі/легітимні»; ML допомагає виявляти нові родини загроз без ручного створення сигнатур.
- **IDS / IPS** — виявлення аномалій у мережевому трафіку, автоматичне блокування підозрілих потоків; ML-моделі дозволяють знаходити складні патерни поведінки.
- **Фрод-детекція (фінтех)** — прогнози ймовірності шахрайських операцій і призначення додаткових перевірок.
- **Біометричні системи** — розпізнавання облич, голосу, поведінки користувача.
- **Аналіз логів / SOAR** — автоматичне корелювання подій, пріоритизація інцидентів і підказки для аналітиків.

5. Зв’язок з помилками першого і другого роду

- **False positives (помилки першого роду)** — система вважає легітимний об’єкт шкідливим → зазвичай призводить до порушення доступності (наприклад, блокування сервісів, незручності для користувачів, можливе відключення захисту).
- **False negatives (помилки другого роду)** — система пропускає шкідливе → порушення цілісності й безпеки ресурсів.
  Вартість кожного типу помилки залежить від домену: у антивірусах іноді прийнятніші false positives, у медицині — обидва типи можуть мати серйозні наслідки.

6. Реальні ризики та наслідки

- **Операційні:** «втома від оповіщень», зростання навантаження на SOC, сповільнене реагування, втрата довіри до системи.

- **Безпекові:** пропуск реальних атак, ескалація прав, витік даних, обходи захисних механізмів.
- **Економічні/репутаційні:** простої сервісів, штрафи, втрата клієнтів, судові претензії.

7. Типові вектори атак на ML-системи

- **Data poisoning (отруєння даних):** ін'єкція шкідливих прикладів у тренувальний чи адаптаційний потік, що змінює параметри моделі в бажаному напрямку.
- **Evasion (обхід класифікатора):** модифікація вхідних прикладів так, щоб обійти детектор (адверсарні збурення в зображеннях/аудіо, мінімальні зміни в мережевих пакетах).
- **Model extraction / inversion:** відновлення або крадіжка моделі/даних через запити до API.
- **Alert flooding / false-positive flood:** створення великої кількості підозрілих, але валідних подій для виклику масових помилок і втоми операторів.
- **Targeted drift:** поступове подання спеціально сформованих валідних прикладів для зміщення розподілу і «приучування» моделі до нових шаблонів.

8. Практичні заходи захисту

1. **Поєднання rule-based підходів і ML.** Поєднання правил і ML дозволяє зменшити вплив разових вразливостей.
2. **Контроль якості даних (data provenance).** Необхідно забезпечувати фільтрацію, автентифікацію джерел телеметрії та валідацію навчальних наборів.
3. **Robust/adversarial training.** Необхідно застосовувати навчання з врахуванням адверсарних прикладів і регулярні stress-тести.
4. **Моніторинг дрейфу (distribution drift) та health-метрик.** Необхідно автоматично відстежувати зміни розподілу ознак, calibration і confidence-метрики; при виявленні аномалій повинні спрацьовувати тригери для розслідування й відкату.

5. **Phased / canary deployment і human-in-the-loop.** Необхідно використовувати поступове розгортання на обмеженому сегменті і забезпечувати можливість ручної перевірки критичних кейсів.
6. **Ансамблі та механізми консенсусу.** Необхідно застосовувати кілька моделей або додаткові перевірочні механізми для зниження ризику уразливості однієї моделі.
7. **Лімітування швидкості подачі даних і квоти.** Необхідно обмежувати потік від одного джерела, щоб ускладнити масовані ін’єкції.
8. **Аудит, логування та forensic-записи.** Необхідно вести докладні логи змін у моделі, джерелах даних та рішеннях для подальшого аналізу.
9. **Red-teaming і зовнішні перевірки.** Необхідно регулярно проводити незалежні тести на пошук worst-case входів і вразливостей.
10. **Навчання персоналу та процеси реагування.** Необхідно впровадити чіткі процедури, SLA з безпеки моделі та інструкції для аналітиків.

9. Архітектурні рекомендації

- **Необхідно проектувати** ML-сервіси за принципом **security-by-design**: передбачати, що дані можуть бути ворожими.
- **Необхідно впроваджувати** кілька рівнів захисту: ML + rule-based + manual review для критичних спрацьовувань.
- **Необхідно встановити** пороги деградації й механізми автоматичного відкату на випадок аномалій.
- **Необхідно регулярно проводити** перевірки та тестування на адаптаційні і адверсарні сценарії.

Систематизація атак за способом впливу, за порушенням безпеки та за специфікою цілей допомагає чітко аналізувати ризики й підбирати відповідні заходи захисту. Машинне навчання — потужний інструмент для безпеки, але водночас відкриває нові вектори загроз: самі дані та процес навчання стають полем бою. Ефективний захист потребує поєднання технічних рішень (робастне навчання, моніторинг, фільтрація), архітектурних підходів (canary, ensemble, лімітування) та організаційних політик (аудит, red-teaming, навчання персоналу).

Наразі забезпечити надійний захист моделей машинного навчання від шкідливих атак набагато складніше, ніж здійснити саму атаку. Це зумовлено тим, що незалежно від обсягу та якості тренування завжди існуватиме певний піднабір даних, на яких модель працюватиме найгірше. Досягти абсолютної точності в 100% неможливо, проте існують підходи, які дозволяють зробити модель більш стійкою до навмисно підготовлених прикладів.

Найважливіша рекомендація полягає в тому, що якщо є можливість уникати використання моделей машинного навчання у середовищах із високим рівнем ризику атак, краще цього дотримуватись. Використання ML цілком виправдане у завданнях з низькою критичністю, наприклад, при класифікації зображень або генерації розважального контенту, де навіть навмисні атаки не спричинять значних соціальних чи економічних наслідків. Проте у випадках, коли система виконує життєво важливі функції, зокрема відповідає за медичну діагностику, виявлення атак на промислові об’єкти або керування безпілотними транспортними засобами, наслідки компрометації можуть бути катастрофічними.

Якщо звернутися до базової задачі класифікації, яка передбачає побудову гіперплощини для розділення простору на класи, то можна побачити внутрішнє протиріччя. У двовимірному просторі це завдання виглядає як пошук межі між двома множинами об’єктів. З одного боку, мета полягає у знаходженні такої функції, яка максимально точно відокремить об’єкти різних класів. З іншого боку, побудувати абсолютно точну лінію неможливо, адже зазвичай у розпорядженні є лише вибірка, а не вся генеральна сукупність даних. Тому доводиться шукати компромісну функцію, яка мінімізує ймовірність помилок класифікації. Інакше кажучи, постає завдання збалансувати прагнення до максимальної точності з необхідністю уникати перенавчання на конкретних прикладах і водночас передбачати поведінку на невідомих даних.

Недонавчання зазвичай виникає, коли модель занадто проста й не здатна врахувати всі закономірності, і найчастіше розв’язується шляхом підвищення складності архітектури. Перенавчання, навпаки, призводить до того, що модель добре працює на тренувальних даних, але погано узагальнює нові приклади. Для

боротьби з цим застосовують методи регуляризації, наприклад L1/L2-нормалізацію або dropout. Такі техніки роблять модель стійкішою до випадкових викидів, проте вони не захищають від навмисно створених шкідливих прикладів. Оптимальний баланс досягається тоді, коли модель є достатньо гнучкою, щоб відображати складні патерни, але водночас не втрачає здатності до узагальнення.

Проблема неправильної класифікації шкідливих прикладів виникає через те, що модель зазвичай не бачила їх у навчальній вибірці, тому часто помиляється. Одним із можливих рішень є доповнення тренувальних даних адверсарними прикладами, створеними штучно, щоб система навчилася їх виявляти. Проте згенерувати всі можливі варіанти таких прикладів і домогтися абсолютної точності неможливо, адже завжди доводиться балансувати між перенавчанням і недонавчанням.

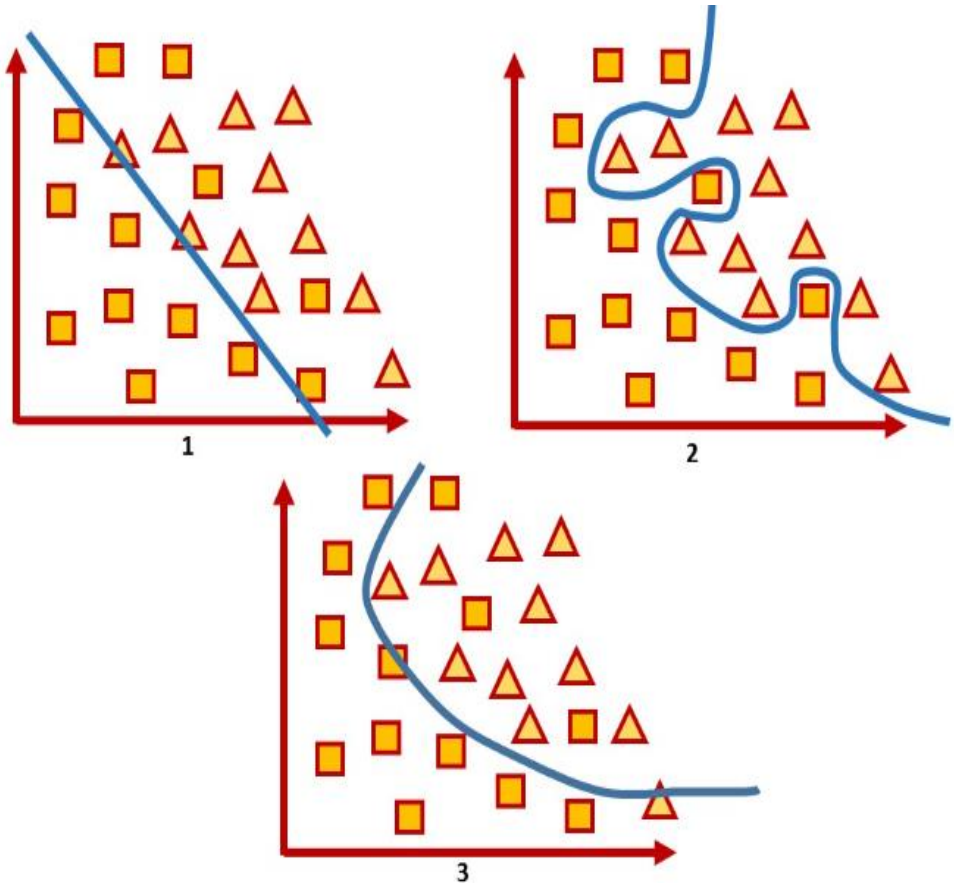


Рис. 1.4 1 - недонавчання, 2 - перенавчання, 3 - оптимальний варіант

Одним із перспективних методів підвищення стійкості моделей є використання генеративно-змагальних нейронних мереж (GAN), що складаються з двох компонентів: генератора та дискримінатора. Генератор створює приклади,

подібні до справжніх, із метою обманути дискримінатор, тоді як дискримінатор навчається відрізняти реальні дані від штучних. У процесі їхньої взаємодії обидві моделі вдосконалюються: генератор навчається створювати все реалістичніші приклади, а дискримінатор — точніше їх виявляти. У результаті формується більш стійка система, яка здатна краще протидіяти шкідливим прикладам.

Однак навіть за умови застосування складних методів захисту завжди існуватиме набір даних, що призведе до некоректних рішень. Це підтверджує фундаментальний факт: моделі машинного навчання ніколи не можуть бути абсолютно невразливими до навмисних атак, тому необхідно орієнтуватися на багаторівневий підхід до безпеки та заздалегідь враховувати потенційні ризики під час створення критичних систем.

Якими можуть бути потенційні наслідки використання машинного навчання з точки зору безпеки?

Дискусії щодо відповідальності систем штучного інтелекту за помилки та небажані соціальні ефекти точаться вже не один рік. Питання ускладнюється тим, що процес створення та впровадження моделей машинного навчання включає декілька груп учасників, і кожна з них впливає на кінцевий результат. З одного боку — це розробники алгоритмів, які обирають методи, налаштовують параметри й формують програмний код. З іншого боку — постачальники даних, адже саме якість, повнота й репрезентативність даних визначають, якою стане модель. І нарешті — кінцеві користувачі або власники системи, які приймають рішення про її експлуатацію у реальному середовищі.

На перший погляд може здаватися, що головна відповідальність лежить на розробнику. Вибір алгоритму, архітектури, методу оптимізації, параметрів регуляризації чи навіть обсягів вибірки для тренування формує базу роботи моделі. Але насправді розробник здебільшого створює програмний продукт, що має відповідати технічному завданню. Як тільки модель демонструє прийнятну точність на тестових даних, її розробка офіційно завершується. Далі починається етап експлуатації, де можуть проявитися проблеми, які неможливо було передбачити під час розробки.

Однією з головних причин є те, що в процесі навчання не використовується вся генеральна сукупність даних. Модель бачить лише обмежену вибірку, тому в реальному середовищі стикається з прикладами, яких раніше не зустрічала. Це може призвести до появи несподіваних, а іноді й небезпечних поведінкових патернів. Іншою причиною є сама природа даних — вони можуть містити упередження, токсичність чи небажані патерни, які система автоматично відтворює.

2 МЕТОДИ ЗАХИСТУ ВІДКРИТОЇ ІНФРАСТРУКТУРИ

2.1 Еволюція архітектур віртуальних мереж

CISC-процесори

У 1970–1980-х роках на ринку мікроелектроніки провідне місце займали CISC-процесори (Complex Instruction Set Computer — комп’ютер із розширеним набором команд), серед яких архітектура x86 стала фактично стандартом для багатьох персональних комп’ютерів та серверних рішень. Основна ідея таких процесорів полягала у виконанні складних операцій за допомогою однієї інструкції, довжина якої могла коливатися від 1 до 15 байт. Інструкція у цьому контексті — це закодована команда, що визначає конкретну дію, яку необхідно виконати процесору.

На перший погляд такий підхід здавався ефективним, адже дозволяв зменшити кількість викликів інструкцій для реалізації складних обчислень. Проте практика показала, що він мав низку суттєвих недоліків:

- **Висока складність апаратної реалізації.** Довгі та варіативні інструкції вимагали більшої кількості транзисторів для декодування та виконання, що значно збільшувало енергоспоживання. Це особливо негативно позначалося на системах, які потребували високої продуктивності при обмежених енергетичних ресурсах, зокрема у сфері 3D-моделювання та рендерингу, що активно розвивалась у той час.
- **Вартість пам’яті та зниження швидкодії.** Для обробки складних інструкцій процесору потрібна була значна кількість оперативної пам’яті. В умовах 1970-х років, коли один мегабайт пам’яті коштував близько 5000 доларів, це ставало критичним обмеженням. Крім того, кожне звернення декодера до пам’яті уповільнювало загальну швидкість виконання інструкцій.
- **Низька ефективність використання командного набору.** Багато можливостей CISC-архітектури, наприклад робота з рядками чи спеціальні комбіновані операції, на практиці застосовувалися рідко. Водночас вони

суттєво ускладнювали структуру декодера, збільшуючи його розмір, енергоспоживання та ймовірність появи помилок.

З розвитком інформаційних технологій, зокрема появою віртуальних мереж і масовим переходом до розподілених обчислень, недоліки CISC-процесорів почали відчуватися ще більше. Для віртуалізованих середовищ потрібні були процесори, здатні швидко й ефективно виконувати однотипні інструкції з мінімальними затримками, забезпечуючи стабільну роботу великої кількості віртуальних машин і мережевих функцій. Складність і різнорідність команд у CISC робила такі завдання менш оптимальними, оскільки затримки на декодування та додаткові витрати енергії негативно впливали на продуктивність віртуальних дата-центрів і мережевих інфраструктур.

Виникнення архітектури RISC

У середині 1980-х років на ринку мікропроцесорів відбулася революція завдяки появі нового архітектурного підходу — RISC (Reduced Instruction Set Computer — обчислювач із скороченим набором команд). Ця концепція принципово відрізнялася від попередньої архітектури CISC і була орієнтована на спрощення інструкцій та підвищення ефективності їхнього виконання.

На відміну від CISC-процесорів, де інструкції мали змінну довжину й могли містити складні комбіновані дії, архітектура RISC базувалася на кількох ключових принципах:

- **Фіксована довжина інструкції** — 4 байти. Це дозволяло значно спростити декодування команд і знизити енергоспоживання, що було особливо важливим для систем, які вимагали високої продуктивності та стабільності.
- **Скорочений набір інструкцій**. З архітектури вилучалися команди, які практично не використовувалися на практиці, але ускладнювали роботу процесора. Завдяки цьому декодер працював швидше, а конструкція процесора ставала простішою.
- **Розширене використання регістрів**. У процесорах RISC було передбачено більшу кількість регістрів, що давало змогу зберігати проміжні дані без частих звернень до оперативної пам’яті. Це не лише прискорювало

обчислення, а й знижувало вартість експлуатації, оскільки у 1980-х роках пам’ять залишалася надзвичайно дорогою.

Найважливішим нововведенням архітектури RISC стала конвеєризація — здатність виконувати кілька інструкцій паралельно. Замість того, щоб чекати завершення попередньої інструкції, як у CISC, процесор міг одночасно перебувати на різних етапах обробки кількох команд. Класична модель конвеєра включала 5 етапів:

1. **Fetch** (зчитування інструкції).
2. **Decode** (декодування).
3. **Execute** (виконання).
4. **Memory Access** (доступ до пам’яті).
5. **Writeback** (запис у регістр).

Кожен із цих етапів займав лише один такт. Таким чином, якщо у CISC три інструкції виконувались послідовно за 15 тактів, то у RISC вони могли бути завершені вже за ті ж 5 тактів завдяки паралельному виконанню. Це забезпечувало суттєвий приріст швидкодії та ефективності.

Завдяки своїм перевагам процесори на основі RISC стали значно продуктивнішими за аналоги на CISC і поступово почали витісняти їх у багатьох сферах. Важливою перевагою було те, що архітектура RISC виявилася особливо ефективною у середовищах, де критичною була масова паралельна обробка даних — наприклад, у віртуальних мережах, які стали активно розвиватися в кінці XX та на початку XXI століття.

У віртуалізованих середовищах і сучасних дата-центрах виникла потреба в обробці тисяч і навіть мільйонів однотипних інструкцій, пов’язаних із маршрутизацією пакетів, підтримкою мережевих протоколів та обробкою запитів від віртуальних машин. Завдяки простим і коротким інструкціям процесори RISC забезпечували низькі затримки та високу ефективність при виконанні повторюваних завдань, що зробило їх ідеальними для використання у системах віртуальних мереж.

Поширення RISC-архітектур

Подальша еволюція мікропроцесорних технологій показала, що підхід RISC став фундаментом для створення цілого покоління процесорів, які суттєво вплинули на розвиток комп'ютерної індустрії та мережевих технологій. У другій половині XX століття на базі RISC було розроблено низку архітектур, які згодом отримали широке поширення та стали важливими елементами інфраструктури цифрового світу.

Найбільш відомою та масово застосованою є архітектура ARM, яка сьогодні фактично стала стандартом у сфері мобільних пристроїв. Процесори ARM використовуються у смартфонах, планшетах, сучасних ноутбуках, а також у персональних комп'ютерах Apple. Популярність ARM пояснюється поєднанням високої продуктивності та енергоефективності, що робить цю архітектуру ідеальною для компактних і мобільних рішень, а також для середовищ, де важливими є низькі тепловиділення та тривала автономна робота.

Ще однією важливою віхою стала архітектура MIPS, яка хоч і втратила свою актуальність у сучасному виробництві, проте залишила значний слід в історії. Саме на базі MIPS створювалися процесори для ігрових консолей PlayStation 1 та PlayStation 2, що демонструє універсальність підходу RISC та його здатність забезпечувати високу продуктивність у спеціалізованих сферах.

У 2010 році з'явилася нова архітектура — RISC-V, яка стала справжнім проривом завдяки своїй відкритості. Її ключовою особливістю є можливість персоналізації: розробники можуть не лише використовувати базовий набір команд, а й додавати власні інструкції, змінювати декодер, підключати опціональні розширення та впроваджувати додаткові механізми захисту. На відміну від ARM, де ліцензування є обов'язковою умовою використання, RISC-V є безкоштовною для застосування і розвитку. Хоча за рівнем оптимізації й зрілості вона у багатьох випадках поступається ARM, її відкритий характер робить архітектуру надзвичайно привабливою для експериментів, досліджень і впровадження у вузькоспеціалізовані системи.

Особливої уваги заслуговує вплив RISC-архітектур на розвиток віртуальних мереж та хмарних середовищ. У таких інфраструктурах важливо забезпечити

баланс між високою продуктивністю, енергоефективністю та гнучкістю налаштувань. ARM-процесори завдяки своїй енергоощадності стали основою для великої кількості дата-центрів, оптимізованих під масштабовані сервіси. Водночас RISC-V відкриває нові можливості: створення спеціалізованих процесорів, налаштованих для конкретних завдань віртуалізації, обробки трафіку чи захисту даних. Персоналізація інструкційного набору дозволяє оптимізувати роботу з потоками даних у віртуальних мережах, мінімізувати затримки та підвищувати рівень безпеки без значних витрат ресурсів.

2.2 Захист пам’яті в архітектурах

Захист пам’яті в архітектурі RISC-V

Архітектура RISC-V передбачає надійний механізм контролю доступу до пам’яті, який отримав назву PMP (Physical Memory Protection). Його головне призначення полягає в тому, щоб на рівні фізичної пам’яті визначати ділянки, доступ до яких може бути суворо обмежений. Налаштування відбувається через спеціальні регістри: у них записується інформація про межі області та права доступу, що дозволяє обмежувати читання, запис або виконання коду. Завдяки цьому програми, що працюють у користувацькому режимі, не можуть отримати доступ до пам’яті, яка має бути недоторканою.

Ключова відмінність PMP від подібних механізмів в архітектурах x86 чи ARM полягає у тому, що він працює не з віртуальною, а саме з фізичною пам’яттю. Це дозволяє забезпечувати більш прямий і передбачуваний рівень захисту, оскільки контроль здійснюється без проміжних рівнів трансляції адрес. Як наслідок, у визначених ділянках фізичної пам’яті захист реалізується жорсткіше й надійніше. Водночас існує і суттєве обмеження: через відносно невелику кількість регістрів (зазвичай від 8 до 16) PMP не підходить для складних багатозадачних операційних систем, таких як Linux. У таких системах необхідно захищати десятки, а то й сотні областей пам’яті, що виходить за межі можливостей PMP. Тому механізм знайшов найефективніше застосування у вбудованих пристроях, IoT-системах та

спеціалізованому обладнанні, де немає повноцінної ОС і кількість критичних ділянок для захисту обмежена.

Принцип роботи PMP ґрунтується на використанні двох типів регістрів:

- Регістрів pmpaddr0–pmpaddrN, які визначають межі захищених ділянок пам’яті. Їхня кількість залежить від конкретної реалізації апаратури, найчастіше — від 8 до 16. Формат адреси задається вибраним режимом адресації: TOR (Top Of Range) або NAPOT (Naturally Aligned Power Of Two), що дозволяє гнучко формувати захищені сегменти.
- Регістрів pmpcfg0–pmpcfgN, які задають права доступу до кожної ділянки. Кожен pmpcfg складається з 8 біт, де окремо визначаються дозволи на читання (R), запис (W) та виконання (X), а також спеціальний біт L (lock), що унеможливлює зміну параметрів доступу до перезапуску системи.

Правильне налаштування PMP має критично важливе значення: помилки у визначенні меж областей чи встановленні дозволів можуть призвести до витоку даних, появи вразливостей або навіть до втрати контролю над системою. З іншого боку, грамотне використання цього механізму дозволяє ефективно ізолювати критично важливі модулі, захистити конфіденційні дані та знизити ризики атак, що експлуатують пам’ять.

Особливу актуальність PMP набуває у середовищах віртуальних мереж та хмарних платформ. Тут він може використовуватися як базовий рівень захисту для обмеження доступу до фізичних ресурсів у багатокористувацьких системах. Завдяки можливості апаратного контролю PMP дозволяє створювати додаткові ізоляційні бар’єри між віртуальними машинами, контейнерами чи мережевими функціями. Це особливо важливо у розподілених системах, де безпека залежить не лише від програмного рівня, а й від апаратних механізмів контролю доступу.

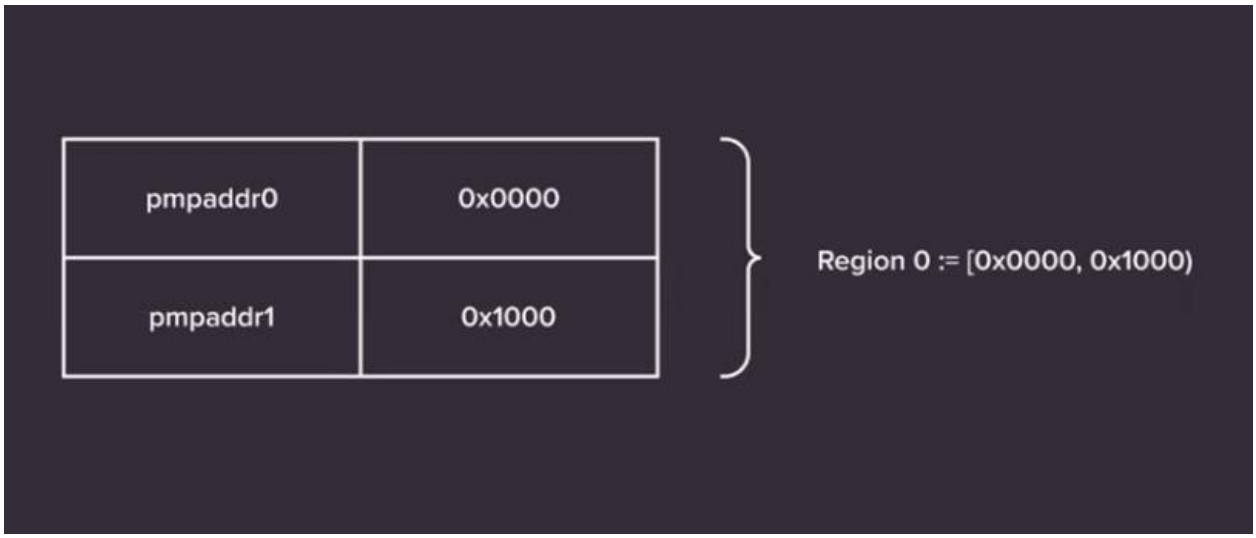


Рис. 2.1 Приклад запису довжини області, що захищається в регістр pmpaddr

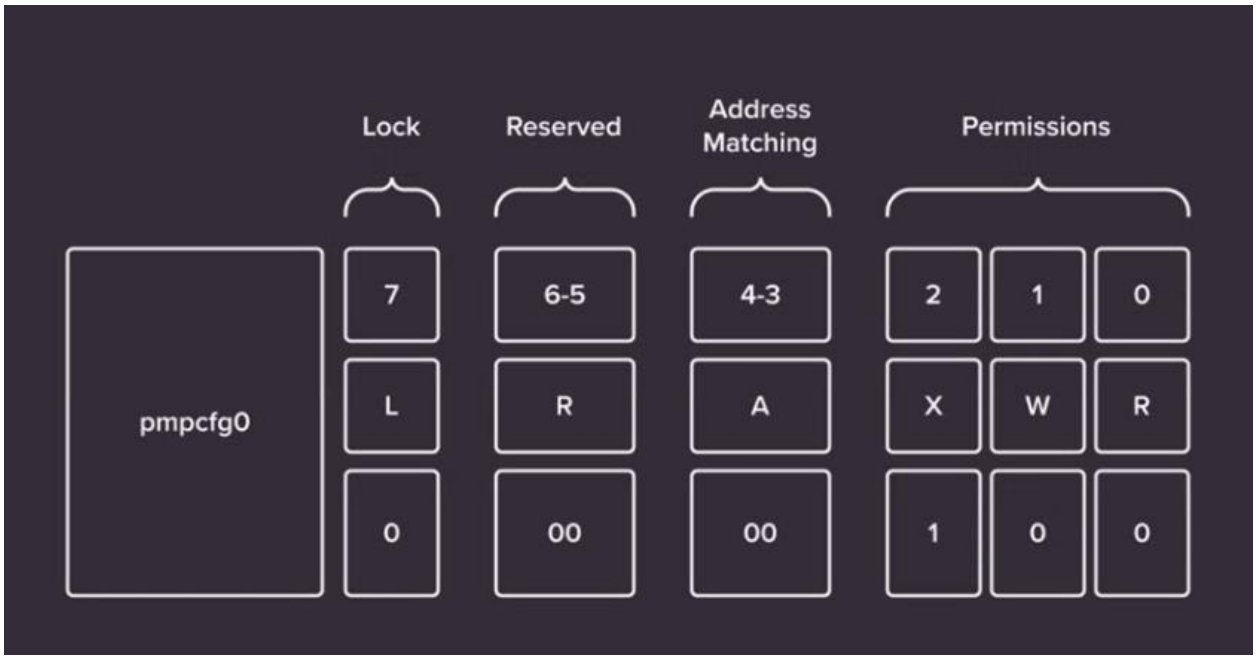


Рис. 2.2 Схема запису байтів у регістр pmpcgf0

Таким чином, PMP в архітектурі RISC-V є ефективним рішенням для систем, де потрібна апаратна ізоляція та мінімальні накладні витрати. Незважаючи на певні обмеження масштабованості, він забезпечує високий рівень безпеки у вбудованих рішеннях, IoT-пристроях, а також може стати основою для побудови надійних інфраструктур у сфері віртуальних мереж і хмарних сервісів, де конфіденційність і цілісність даних мають вирішальне значення.

Захист пам’яті в архітектурі x86

В архітектурі x86 захист пам’яті побудований на принципово іншій концепції, ніж у RISC-V, — він реалізується через механізм сторінкової організації (paging),

що базується на застосуванні віртуальної пам’яті. Цей механізм отримав назву Paging Protection і став одним із ключових елементів, які забезпечують стабільність і безпеку сучасних обчислювальних систем.

Основні завдання цього механізму такі:

- **Ізоляція процесів.** Кожному процесу надається власний віртуальний адресний простір, що унеможливлює прямий доступ до пам’яті іншого процесу. Це захищає як операційну систему, так і прикладні програми від випадкових чи навмисних втручань.
- **Контроль доступу на рівні сторінок.** Пам’ять поділяється на сторінки стандартного розміру (зазвичай 4 КБ), для кожної з яких можна встановити права доступу: читання, запис, виконання. Завдяки цьому можливо реалізувати точковий захист і більш гнучке керування ресурсами.
- **Механізм кільцевих рівнів (Ring levels).** Архітектура x86 підтримує 4 кільця привілеїв (Ring 0–3), що дозволяє розмежувати права доступу між ядром ОС, драйверами, системними службами та прикладними програмами. Таким чином створюється багаторівнева модель безпеки, де найбільш критичні функції доступні лише на найнижчому рівні привілеїв.

Важливою особливістю сторінкового захисту є його багаторівнева структура. Для управління пам’яттю використовуються складні таблиці відображення адрес:

- **PML4 (Page Map Level 4)** — найвищий рівень у 64-бітних системах, що дозволяє опрацьовувати великі адресні простори;
- **PDP (Page Directory Pointer)** — вказівники на каталоги сторінок;
- **PD (Page Directory)** — каталоги, які об’єднують записи таблиць сторінок;
- **PT (Page Table)** — таблиці сторінок, що визначають відповідність між віртуальною та фізичною адресою.

Завдяки цій ієрархії можна ефективно керувати пам’яттю у складних операційних системах, оптимізувати її використання та забезпечувати захист від несанкціонованого доступу. Система paging дозволяє ізолювати сотні чи навіть тисячі процесів у багатозадачному середовищі, що особливо важливо для серверів, дата-центрів та хмарних платформ.

Водночас цей механізм має і свої недоліки. Його налаштування та обслуговування досить складні, що збільшує накладні витрати при виконанні операцій. Крім того, перехід між віртуальними й фізичними адресами створює додаткові затримки, що не є критичними для повноцінних ОС, але може стати проблемою для простих вбудованих систем або пристроїв IoT. Саме тому сторінкова організація пам’яті у x86 чудово підходить для складних операційних середовищ, проте є надмірною для компактних і ресурсно обмежених систем.

У контексті віртуальних мереж і хмарних обчислень переваги paging стають особливо помітними. Завдяки багаторівневій організації можна створювати надійну ізоляцію між віртуальними машинами, забезпечувати сегментацію пам’яті в контейнерних середовищах і реалізовувати додаткові інструменти безпеки, що перешкоджають витоку даних або несанкціонованому доступу до ресурсів. Саме тому механізм Paging Protection є фундаментальним елементом архітектури x86, без якого неможливо уявити сучасні системи керування пам’яттю у віртуалізованих і розподілених інфраструктурах.

Порівняння Keystone Enclave з аналогами інших архітектур

Одним із найбільш відомих рішень у сфері захищених обчислень є Intel SGX (Software Guard Extensions) — набір інструкцій для архітектури x86, розроблений компанією Intel. Його головна ідея полягає у створенні спеціальних захищених середовищ виконання — так званих анклавів, які розташовуються всередині адресного простору одного процесу. На відміну від Keystone, що є більш гнучкою та відкритою платформою, SGX створює анклав, жорстко обмежений одним процесом, проте забезпечує його апаратне шифрування та ізоляцію від усієї решти операційної системи, драйверів і навіть від гіпервізора.

Архітектура SGX функціонує на рівні користувацького режиму (Ring 3) і формує анклав у вигляді виділеної захищеної області оперативної пам’яті (DRAM), доступ до якої суворо контролюється на апаратному рівні. Це перетворює SGX на повноцінне середовище довірених обчислень (Trusted Execution Environment, TEE). Завдяки цьому SGX активно використовується у сфері хмарних обчислень, де безпека та конфіденційність даних є критично важливими. Одним із найвідоміших

прикладів є інтеграція SGX у сервіс Azure Confidential Computing від Microsoft, де за допомогою анклавів реалізується захист даних під час їхньої обробки у віртуальних середовищах.

Попри очевидні переваги, SGX має й ряд обмежень. Однією з основних проблем є висока складність розробки програмного забезпечення для цієї технології. Створення застосунків під SGX потребує від розробників глибоких знань внутрішніх механізмів архітектури, що ускладнює як програмування, так і відлагодження. Ще одним суттєвим недоліком є закритість самої технології: SGX є повністю пропрієтарною розробкою Intel, тому аналіз її внутрішніх алгоритмів і виявлення прихованих уразливостей практично неможливі для сторонніх дослідників. Саме через це у SGX неодноразово виявлялися критичні вразливості, усунення яких було складним або взагалі обмеженим через закритий характер архітектури.

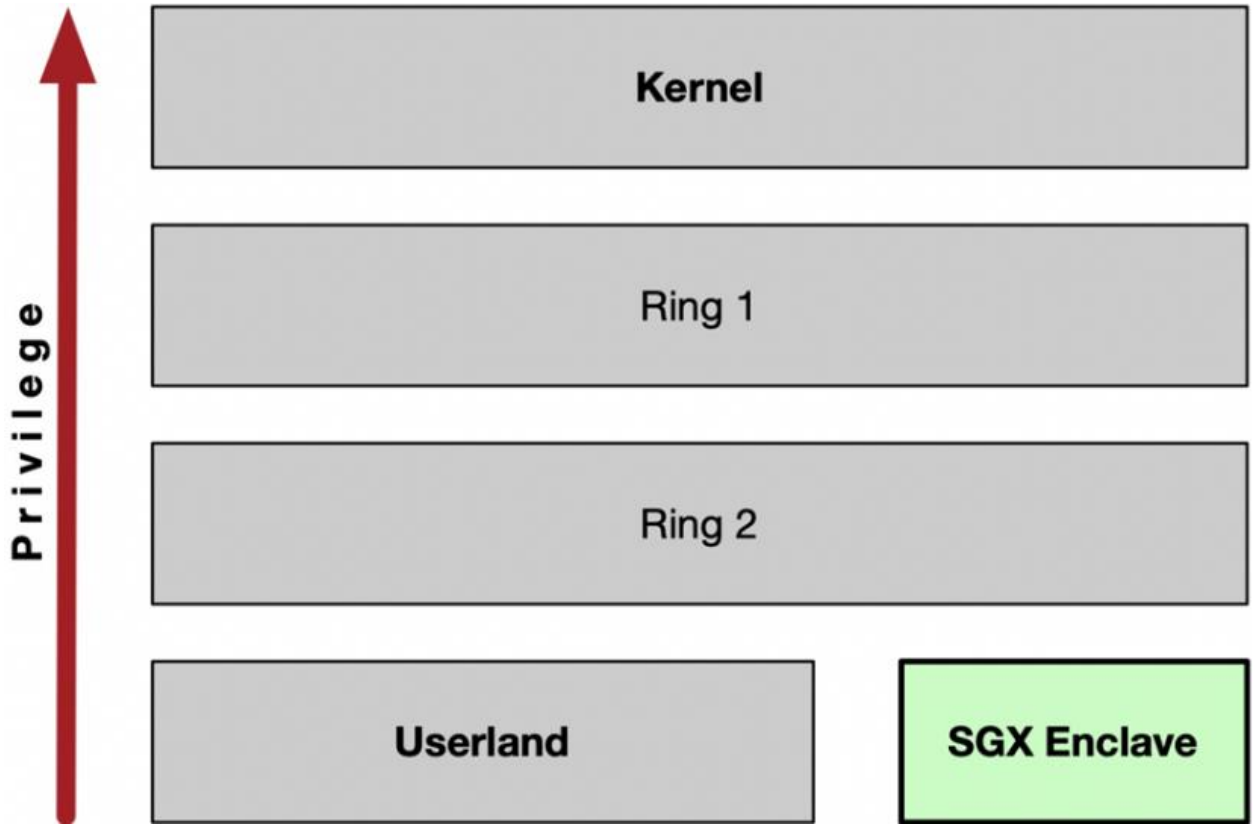


Рис. 2.3 Робота анклаву Intel SGX

У порівнянні з Keystone Enclave, який є відкритим і гнучким рішенням з можливістю кастомізації під різні типи пристроїв, SGX забезпечує високий рівень захисту на апаратному рівні, але обмежений жорсткими рамками пропрієтарності

та складністю у використанні. Keystone, у свою чергу, завдяки відкритій архітектурі та підтримці RISC-V, може адаптуватися для широкого спектра завдань, включно з інтеграцією у віртуальні мережі та хмарні сервіси, де потрібна як безпека, так і гнучкість у налаштуваннях.

TrustZone — це апаратна технологія, розроблена компанією ARM, яка забезпечує поділ системи на два окремі середовища виконання, що отримали назву «світи»:

- **Secure World** — захищене середовище, яке фактично виступає довіреною обчислювальною платформою (**Trusted Execution Environment, TEE**). У цьому режимі виконуються найбільш критичні з точки зору безпеки завдання: робота з криптографічними ключами, перевірка автентичності, шифрування даних, керування платіжними транзакціями тощо.
- **Normal World** — основне середовище, у якому функціонує стандартна операційна система разом із прикладними програмами. Саме воно відповідає за більшість користувацьких завдань і забезпечує звичний функціонал пристрою.

Перемикання між цими «світами» здійснюється за допомогою спеціальних інструкцій SMC (Secure Monitor Call), які дозволяють викликати монітор безпеки. Важливою особливістю є асиметрія доступу: Secure World має змогу отримувати доступ до ресурсів Normal World, тоді як навпаки це неможливо. Така модель дозволяє створити чіткий бар’єр захисту та гарантувати, що критично важливі дані та обчислення залишаються ізольованими від потенційно небезпечного середовища користувацьких програм.

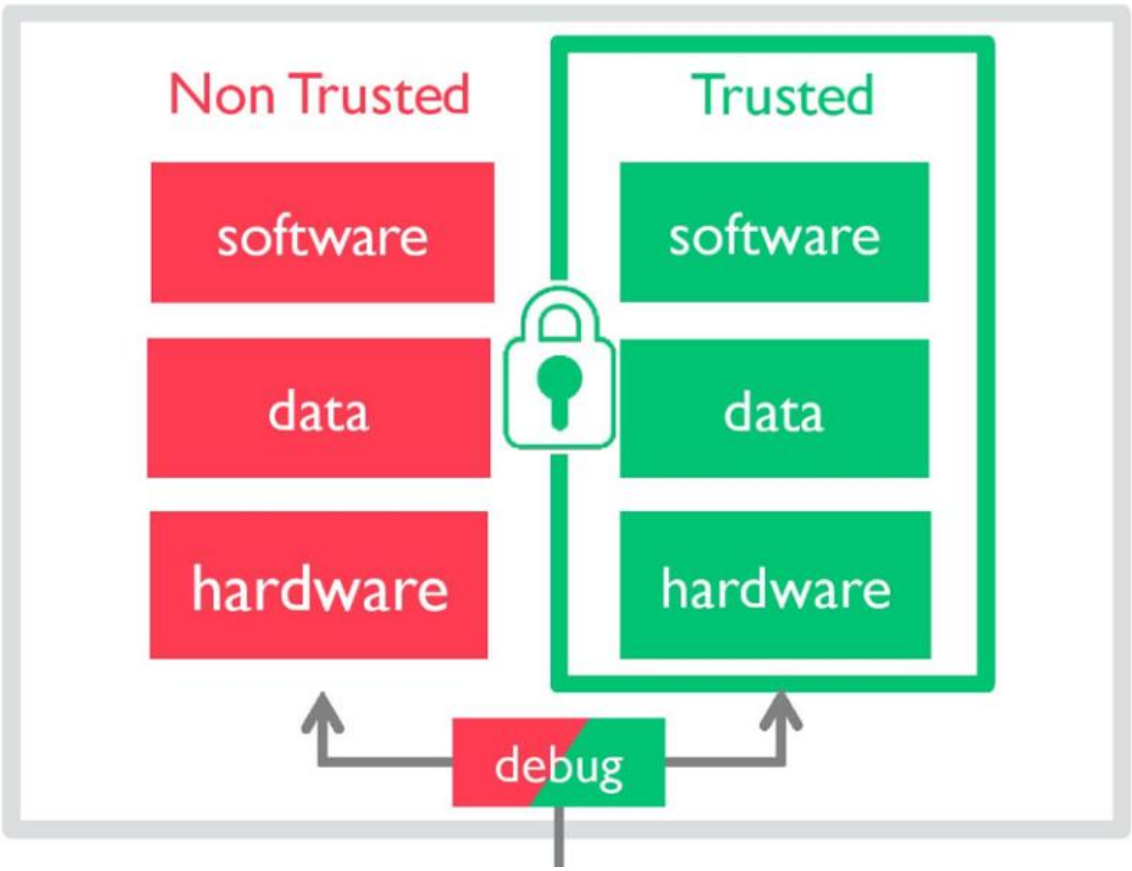


Рис. 2.4 Приклад роботи TrustZone

Технологія TrustZone активно використовується у мобільних пристроях, зокрема в екосистемі Android, для роботи з довіреними операційними системами, наприклад Trusty OS. Крім того, вона часто інтегрується у чипи з підтримкою DRM (Digital Rights Management) — технічних засобів захисту авторських прав, що дозволяє безпечно зберігати та обробляти мультимедійний контент, ліцензії чи цифрові сертифікати. Додатково TrustZone застосовується у таких сферах, як біометрична автентифікація (сканери відбитків пальців, розпізнавання обличчя), мобільні платіжні системи (Google Pay, Samsung Pay) та захищені канали зв’язку.

Однак ця технологія має і свої недоліки. Найбільша проблема полягає у закритості реалізації: внутрішні механізми TrustZone є пропрієтарними, тому незалежні розробники чи дослідники не мають повного доступу до архітектури. Це обмежує можливість гнучкої адаптації TEE під специфічні завдання та ускладнює створення нових рішень у наукових або експериментальних проєктах. Крім того, у зв’язку з високою залежністю від виробника апаратного забезпечення, складно

забезпечити однаковий рівень безпеки у різних пристроях, навіть якщо вони побудовані на однаковій ARM-архітектурі.

У сучасних умовах, коли зростає значення віртуальних мереж і хмарних сервісів, TrustZone також може відігравати ключову роль. Його використання дозволяє створювати безпечні домени для ізоляції віртуальних машин, захищати конфіденційні дані у хмарних середовищах та забезпечувати контроль доступу на рівні апаратури. Наприклад, у розподілених мережах TrustZone може використовуватися для захищеної обробки ключів шифрування чи аутентифікації користувачів у багатокористувацьких віртуалізованих системах.

2.3 Контроль захисту в процесорах віртуальних мереж

Контроль привілеїв: значення та особливості реалізації

У сучасних процесорах контроль привілеїв є одним із ключових механізмів, що забезпечує стабільність та безпеку обчислювальних систем. Його суть полягає у багаторівневому розмежуванні доступу до ресурсів: кожен рівень визначає, які команди можуть виконуватися та які ділянки пам’яті чи апаратні компоненти є доступними. Завдяки цьому досягається апаратна ізоляція — користувацькі програми не здатні безпосередньо впливати на роботу ядра системи чи апаратури. Відсутність такого контролю неминуче призвела б до хаотичного доступу до ресурсів, збоїв у роботі пристроїв і потенційних загроз безпеці.

Контроль привілеїв потрібен не лише для запобігання втручанню шкідливих програм у критично важливі процеси, але й для організації багатозадачності та віртуалізації. У складних інфраструктурах, наприклад у хмарних дата-центрах чи віртуальних мережах, десятки й сотні процесів виконуються одночасно, і тільки жорстке розмежування доступу дозволяє гарантувати, що один процес не зможе вплинути на інший або отримати конфіденційні дані.

Механізм привілеїв у RISC-V

В архітектурі **RISC-V** реалізовано три рівні привілеїв, що відрізняються своїм функціональним призначенням:

- **Machine mode (M-mode)** — найвищий рівень, що має повний доступ до всіх апаратних ресурсів, включно з фізичною та віртуальною пам’яттю. Саме в цьому режимі працюють низькорівневі прошивки, механізми завантаження та системи ініціалізації. Він відповідає за управління апаратними компонентами, а також за контроль над нижчими рівнями.
- **Supervisor mode (S-mode)** — середній рівень, який використовується операційними системами. Тут реалізується управління процесами, робота з віртуальною пам’яттю, системними викликами та міжпроцесною взаємодією. Цей рівень забезпечує основу для роботи багатозадачних ОС.
- **User mode (U-mode)** — найнижчий рівень, у якому виконуються прикладні програми. Доступ до апаратури тут суворо обмежений: усі привілейовані дії можливі лише через системні виклики, що проходять перевірку на вищих рівнях.

Така трирівнева ієрархія дозволяє створити чіткий поділ між прикладними програмами, операційною системою та апаратним забезпеченням, знижуючи ризик несанкціонованого доступу чи порушення роботи системи.

Порівняння з архітектурою x86

У x86 реалізована більш складна модель контролю привілеїв, яка базується на концепції «кілець» (Rings). Тут виділяють чотири рівні:

- **Ring 0** — найвищий рівень, у якому виконується ядро операційної системи. Має повний доступ до апаратних ресурсів і є найбільш привілейованим.
- **Ring 1 і Ring 2** — проміжні рівні, які теоретично призначені для драйверів або системних служб, проте на практиці використовуються рідко.
- **Ring 3** — рівень користувацьких застосунків, які працюють із мінімальними привілеями та не мають прямого доступу до апаратних ресурсів.

Таким чином, хоча x86 формально підтримує чотири кільця привілеїв, у реальних системах активно застосовуються лише два: Ring 0 для ядра ОС і Ring 3 для користувацьких програм.

Практичне значення у віртуальних середовищах

Контроль привілеїв має особливе значення у сфері віртуалізації та віртуальних мереж. У хмарних інфраструктурах, де одночасно виконуються сотні віртуальних машин або контейнерів, ізоляція між рівнями дозволяє гарантувати безпеку та стабільність роботи всієї системи. Наприклад:

- у RISC-V M-mode може використовуватися для управління гіпервізором, тоді як S-mode — для гостьових ОС, а U-mode — для користувацьких процесів;
- у x86 роль віртуалізаційного шару може виконуватися на рівні Ring 0, тоді як гостьові системи обмежуються нижчими привілеями, що гарантує відокремлення їхніх ресурсів.

Таким чином, механізм привілеїв є універсальним інструментом, який забезпечує як безпеку, так і ефективне управління ресурсами в сучасних комп’ютерних системах. В архітектурі RISC-V він реалізований у простій та зрозумілій трирівневій моделі, а в x86 — у більш гнучкій чотирирівневій системі, проте в обох випадках його основна мета залишається незмінною: захист апаратних і програмних ресурсів від небажаних втручань та забезпечення ізоляції віртуальних середовищ.

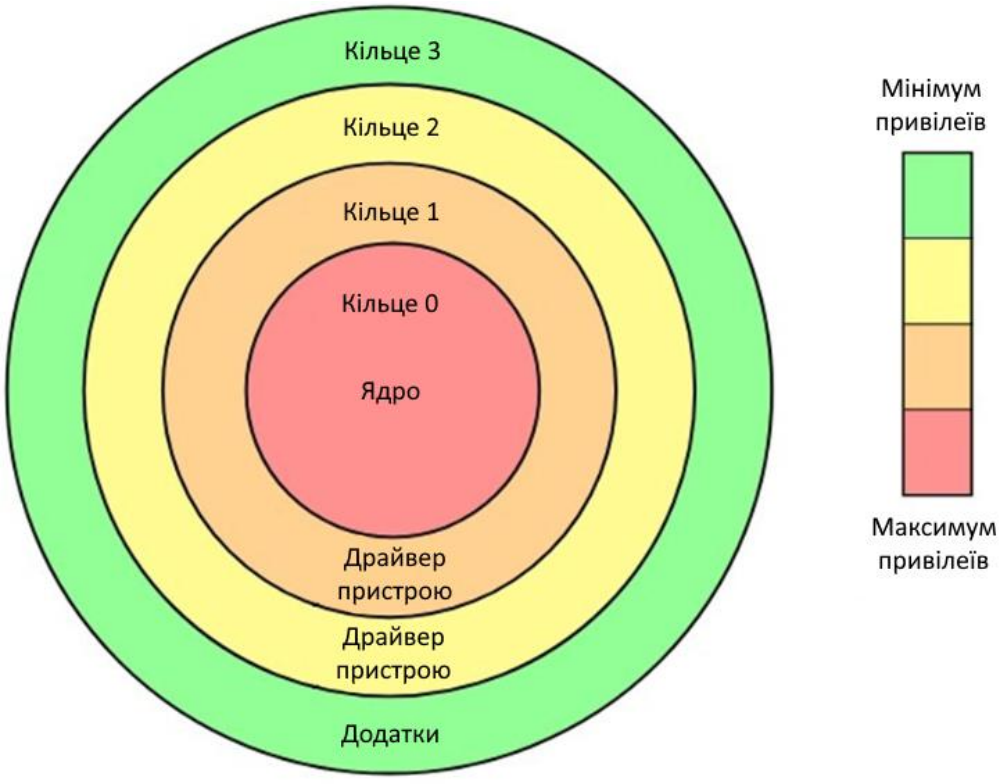


Рис. 2.5 Рівні привілеїв x86

В архітектурі ARM реалізовано власну систему контролю привілеїв, яка еволюціонувала разом із розвитком самої архітектури. У ранніх версіях ARM цей підхід був досить подібним до механізму, що застосовується в RISC-V: система мала кілька рівнів доступу, кожен із яких визначав, які інструкції дозволено виконувати та які ресурси можна використовувати. Такий розподіл забезпечував базову ізоляцію користувацьких програм від операційної системи та апаратних ресурсів.

Однак із часом вимоги до безпеки значно зросли — зокрема через розвиток мобільних технологій, хмарних сервісів і віртуальних мереж, де обробка конфіденційних даних потребувала якісно нового рівня захисту. У відповідь на ці виклики компанія ARM впровадила механізм TrustZone, який став важливим кроком у бік підвищення надійності архітектури.

TrustZone поділяє систему на два незалежні середовища:

- **Normal World** — звичайне середовище, де працює основна операційна система, драйвери та прикладні програми.
- **Secure World** — захищене середовище, яке виконує критично важливі функції: роботу з криптографічними ключами, управління автентифікацією користувачів, перевірку цілісності системи, обробку платіжних операцій тощо.

Перемикання між цими світами здійснюється за допомогою спеціальних інструкцій SMC (Secure Monitor Call), що дозволяють створити надійний бар’єр між звичайним та захищеним просторами виконання. Унікальною особливістю є асиметричний доступ: Secure World може взаємодіяти з ресурсами Normal World, тоді як у зворотному напрямку доступ заборонений. Це забезпечує ефективний контроль і знижує ризики компрометації конфіденційних даних.

Завдяки TrustZone архітектура ARM отримала додатковий рівень безпеки, що дозволяє виконувати важливі завдання ізольовано від основної ОС. Це особливо актуально для мобільних пристроїв, де TrustZone застосовується для запуску довірених операційних систем (наприклад, Trusty OS), управління захищеними

транзакціями, зберігання біометричних даних і роботи з DRM-системами. У ширшому контексті, TrustZone можна розглядати як важливий фундамент для побудови безпечних віртуальних середовищ та хмарних платформ, де потрібна апаратна ізоляція для одночасного виконання численних сервісів.

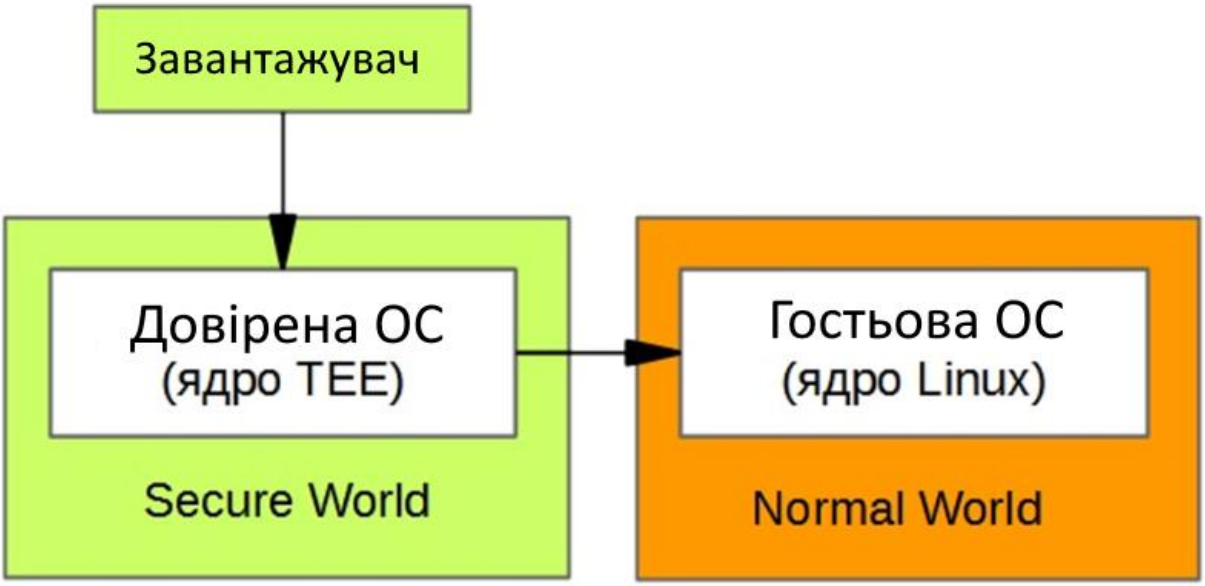


Рис. 2.6 Рівні привілеїв в ARM

Захист від спекулятивних атак у процесорах

Сутність спекулятивного виконання та його вразливості

Сучасні процесори, незалежно від архітектури, активно використовують техніку спекулятивного виконання інструкцій. Її мета полягає у підвищенні продуктивності: процесор починає виконувати інструкції ще до того, як остаточно стане відомим, чи вони потрібні. Це дозволяє зменшити простої під час очікування результатів порівнянь та зробити роботу системи значно швидшою.

Однак у 2018 році було доведено, що ця оптимізація відкрила небезпечний клас атак, які дозволяють отримати доступ до даних, що повинні бути недосяжними для користувацьких програм. Сценарій атаки виглядає так:

1. Зловмисник багаторазово виконує код із коректним індексом **x**, привчаючи процесор спекулятивно обробляти саме цю гілку коду.
2. Потім передається індекс, що виходить за межі масиву та вказує на захищену ділянку пам’яті (наприклад, де зберігаються паролі чи ключі шифрування).

3. Процесор:
    - завантажує значення array[x] спекулятивно,
    - перевіряє умову та виявляє, що доступ заборонений,
    - скасовує операцію, але сліди доступу залишаються у кеші.
4. Зловмисник аналізує час доступу до пам’яті:
    - швидкий доступ свідчить, що дані потрапили у кеш,
    - повільний доступ означає відсутність у кеші.

Таким чином, через побічні канали відновлюється секретна інформація. Найвідоміші уразливості цього класу:

- **Spectre** — маніпуляція передбаченням переходів, що дозволяє зламати ізоляцію між процесами та отримати дані іншого застосунку.
- **Meltdown** — використання спекулятивного виконання для доступу до захищеної пам’яті ядра.

Ці атаки вразили більшість процесорів архітектур x86 та ARM, що спричинило необхідність у серйозних змінах як на апаратному, так і на програмному рівнях.

Методи захисту в RISC-V та інших архітектурах

RISC-V принципово відрізняється від x86 і ARM тим, що для досягнення високої продуктивності ця архітектура не вимагає обов’язкового використання спекулятивного виконання. Проте деякі ядра все ж інтегрують цю технологію, що відкриває можливість для атак.

- Наприклад, **64-бітне ядро CVA6** підтримує спекулятивне виконання, що дозволяє швидше працювати з пам’яттю та підвищує ефективність. Однак водночас з’являється вразливість до атак типу Spectre. Для зменшення ризиків була введена інструкція fence, яка примусово очищує спекулятивні операції під час зміни контексту. Цей метод ефективний проти Spectre-V2, але не забезпечує повного захисту від Spectre-V1, де атака базується на умовних переходах.

- Натомість ядро **Ariane** взагалі не підтримує спекулятивне виконання. Це робить його менш продуктивним у порівнянні з CVA6, проте воно стає повністю невразливим до всього класу атак Spectre/Meltdown.

В архітектурі x86 компанії Intel та AMD пішли шляхом удосконалення існуючих механізмів:

- **IBRS (Indirect Branch Restricted Speculation)** — обмежує спекулятивне виконання непрямих переходів;
- **STIBP (Single Thread Indirect Branch Predictors)** — захищає від атак між потоками, що особливо важливо у віртуальних середовищах і хмарних сервісах.

Такі методи фактично забороняють процесору використовувати історію передбачень переходів, що блокує більшість варіантів Spectre, але робить систему все ще вразливою до Meltdown. Крім того, вони знижують продуктивність на 5–15%. Альтернативним рішенням є бар'єр LFENCE, який примусово зупиняє спекулятивне виконання наступних команд. Він захищає від Spectre, частково від Meltdown, і знижує швидкодію приблизно лише на 5%.

Архітектура ARM виявилася навіть більш чутливою до Spectre, ніж x86. Тому було розроблено аналог LFENCE — інструкцію CSDB, яка блокує більшість атак цього класу, але знижує продуктивність приблизно на 10%. Додатково ARM впровадила апаратний метод SSBS (Speculative Store Bypass Safe), що відключає спекулятивне оминання перевірок пам'яті. Він захищає від значної частини атак Spectre, знижуючи продуктивність у межах 5–10%. Важливо, що цей механізм активується через регістр SCTLR_ELx.SSBS, тоді як за замовчуванням увімкнено лише CSDB.

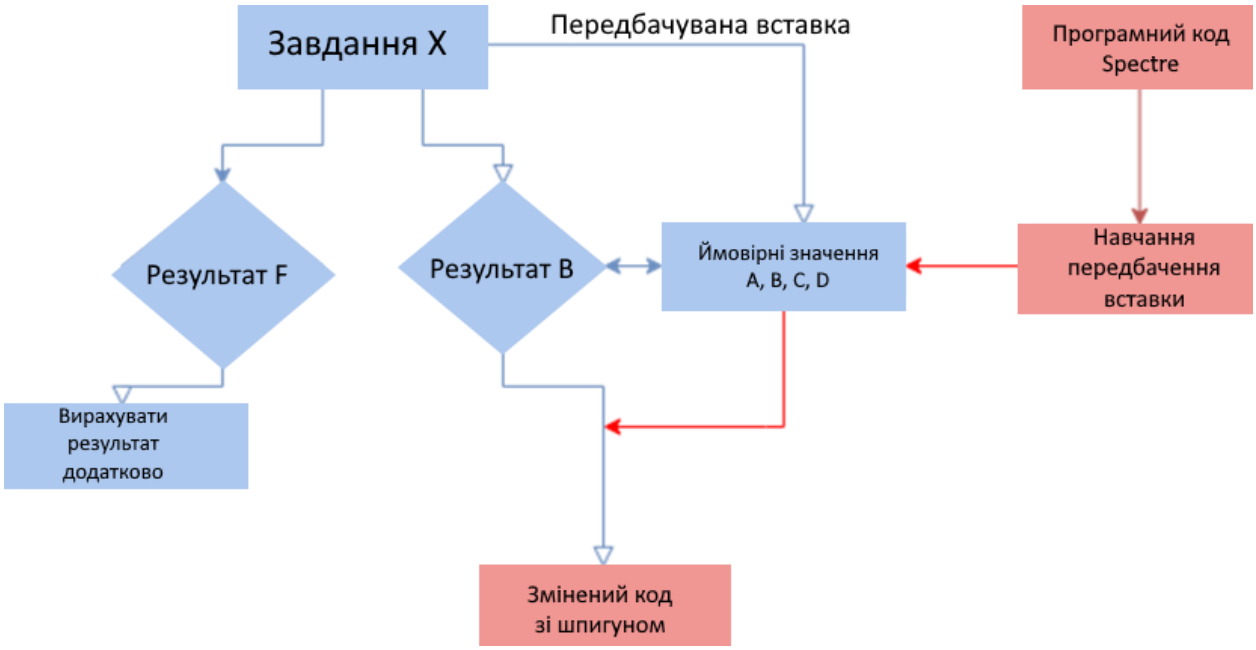


Рис. 2.7 Приклад роботи Spectre-атаки

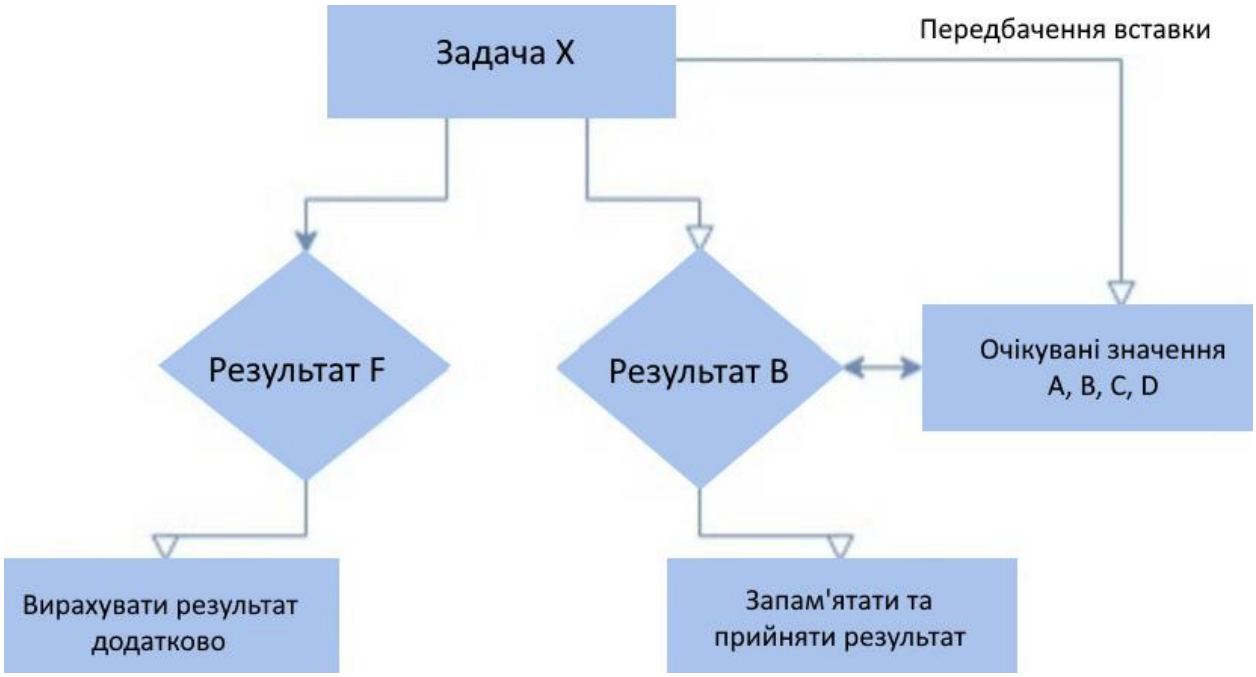


Рис. 2.8 Приклад роботи Spectre-атаки

3 ЗАХИСТ ВІРТУАЛЬНОЇ МЕРЕЖІ L2 КОМУТАТОРАМИ

3.1 Методи захисту

Уся термінологія та конфігураційні команди наведені відповідно до документації компанії **Cisco**, яка фактично використовується як галузевий стандарт. На початку опису кожної атаки міститься короткий екскурс у механізм роботи того протоколу, що піддається атаці. Матеріал розраховано скоріше на початківців, ніж на досвідчених мережевих інженерів. У роботі розглядатимуться такі атаки:

- **Rogue DHCP Server**
- **DHCP starvation**
- **CAM-table overflow**
- **VLAN hopping**
- **MAC-spoofing**

За основу викладу взято відеоурок **CBT Nuggets** із циклу **CCNA Security**.

Rogue DHCP Server

Опис протоколу (спрощено)

Щоб зрозуміти природу атаки, коротко нагадаємо класичну послідовність обміну в DHCP:

1. **Discover** — клієнт без IP надсилає широкомовний DHCPDISCOVER на 255.255.255.255, очікуючи відповіді серверів у мережі.
2. **Offer** — DHCP-сервер(и) відповідають DHCPOFFER, пропонуючи конфігурацію (IP, маска, шлюз, DNS та ін.). Відповідь адресована MAC-адресі клієнта.
3. **Request** — клієнт обирає один із запропонованих варіантів і надсилає DHCPREQUEST, вказуючи обраний сервер.
4. **Acknowledgment** — сервер надсилає DHCPACK; після цього клієнт налаштовує мережеві параметри і може працювати в мережі.

Механіка атаки Rogue DHCP Server. Rogue DHCP Server — це ситуація, коли в мережі з’являється несанкціонований

DHCP-сервер (зловмисний або помилково налаштований), який відповідає на DHCPDISCOVER швидше або агресивніше ніж легітимні сервери. Агресивний/зловмисний сервер може:

- пропонувати клієнтам некоректні значення шлюзу (default gateway) і DNS (переадресація трафіку через контролювані інтерфейси),
- видавати IP-адреси з неправильної підмережі або з діапазону, що конфліктує з існуючими адресами,
- вказувати параметри, що спрямовують трафік через проксі/малварні шлюзи або логують трафік,
- виділяти короткі оренди (lease), змушуючи клієнтів часто переадресовуватись і створюючи підвищене навантаження/нестабільність.

Типовий сценарій: зловмисник піднімає DHCP-сервер на ноутбуці/комутаті або через скомпрометований порт у кімнаті, налаштовує DHCPOFFER з власними параметрами і тим самим перехоплює мережевий трафік користувача (ман-ін-зе-мідл, підміна DNS і т.д.).

Наслідки атаки:

- **Перехоплення трафіку і витік даних** (через підміни DNS/шлюзу);
- **Фішинг і перенаправлення на шкідливі сайти;**
- **Порушення доступності сервісів** через некоректну адресацію або зіткнення IP;
- **Підвищене навантаження на інфраструктуру** (короткі lease, масові переналаштування клієнтів);
- **Зниження довіри користувачів і ризик інцидентів безпеки** у корпоративній мережі.

Як виявити Rogue DHCP Server:

- Моніторинг DHCPOFFER/DHCPACK у мережевому трафіку (ошибка: відповіді від невідомих MAC/IP).
- Логи DHCP-серверів: сплески відмов або нестандартні запити.
- Виявлення змін у параметрах клієнтів (зміна шлюзу/DNS без адмін-ініціативи).

- Використання інструментів сканування мережі та IDS/IPS-сигнатур, які фіксують наявність незареєстрованих DHCP-серверів.
- ARP-/DHCP-сканування на портах доступу до комутаторів (локальні аномалії).

Захист і заходи протидії:

1. **DHCP Snooping** (на комутаторах Cisco і сумісних рішеннях) — ключовий захід:
    - Проводиться маркування інтерфейсів як «trusted» (легітимні DHCP-реле/сервери) або «untrusted» (порт користувача).
    - Комутатор дозволяє пропозиції DHCP лише з trusted портів і відкидає пропозиції з untrusted.
    - Snooping також формує базу DHCP bindings (IP↔MAC↔порт), яка корисна для подальших політик (ARP inspection, IP source guard).
2. **Dynamic ARP Inspection (DAI)** — блокує підроблені ARP-відповіді, ґрунтуючись на DHCP-binding-таблиці; ефективно в парі з DHCP Snooping.
3. **IP Source Guard** — дозволяє блокувати трафік, якщо IP/MAC не відповідають записам із таблиці DHCP Snooping (перешкоджає підміні IP).
4. **Патчинг і сегментація мережі**: організувати VLAN-сегментацію для гостів/IoT, де доступ до DHCP-серверів обмежено.
5. **Аутентифікація портів (802.1X)** — вимагає від пристроїв проходження автентифікації перед наданням мережевого доступу; знижує ризик підключення неавторизованих DHCP-серверів.
6. **Строге управління інфраструктурою**: заборона підключення несертифікованих пристроїв у критичні сегменти; інвентаризація активів.
7. **Моніторинг і реагування**: налаштувати алерти на появу невідомих DHCP-відповідей; автоматичні сценарії ізоляції порту (port shutdown/errdisable) при виявленні підозрілих DHCP-серверів.

Приклад конфігурації (ідея): на Cisco-комутаторі вмикають ip dhcp snooping глобально, потім на інтерфейсі uplink: ip dhcp snooping trust, а на портах

користувачів — за замовчуванням untrusted; додатково — ip source binding / ip dhcp snooping database для централізованого зберігання binding-таблиці.

Рекомендації для адміністраторів:

- Насамперед увімкнути DHCP Snooping у критичних VLAN/сегментах.
- Підключити DAI та IP Source Guard у тих сегментах, де точність адресації критична.
- Регулярно перевіряти DHCP-binding logs і налаштування trusted портів.
- Навчити персонал політикам безпечного підключення обладнання (гостьові мережі, обмежений доступ).
- Використовувати SIEM/IDS для кореляції подій (неочікувані DHCPOFFER, багато DHCPREQUEST від одного порту тощо).

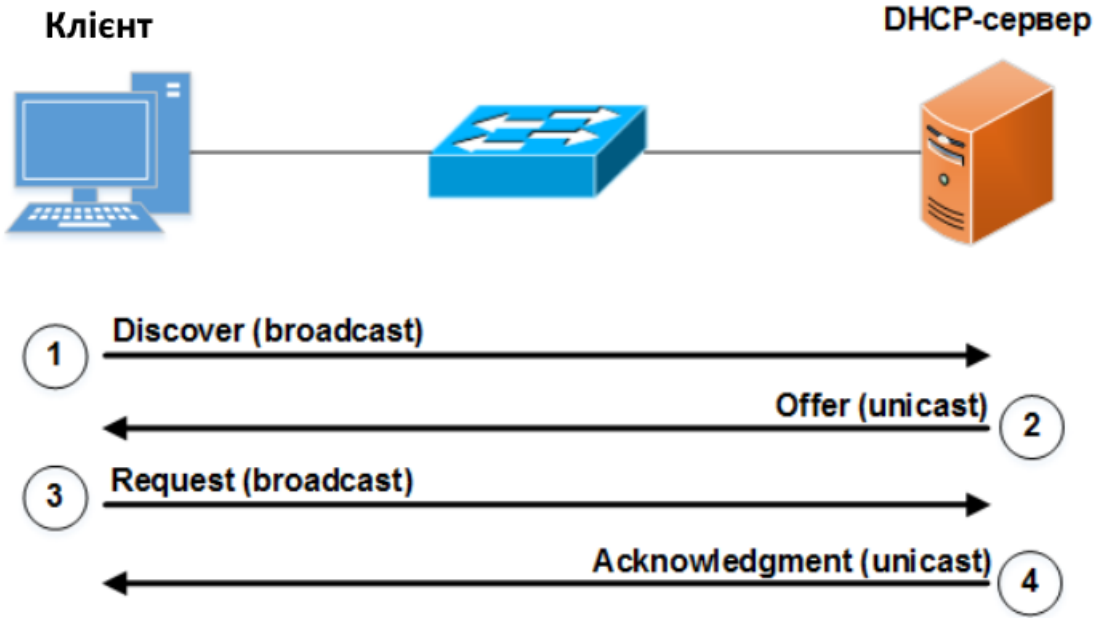


Рис. 3.1 Підміна DHCP-сервера

Мета цієї атаки — підміна легітимного DHCP-сервера фальшивим. Якщо в мережі одночасно працюють два DHCP-сервери, і один із них налаштований зловмисником (або некоректно), частина хостів може отримати від нього некоректні мережеві параметри: IP-адреси, маски, DNS-сервери та, найважливіше, шлюз за замовчуванням.

У результаті підміни шлюзу фейковий DHCP-сервер отримує можливість перенаправляти весь трафік жертв через контрольовану ним інфраструктуру — виконувати прослуховування, логування, зміну або перенаправлення пакетів. Це

дає простий і ефективний спосіб реалізувати атаку типу Man-in-the-Middle: зловмисник може підмінювати DNS-відповіді, підсовувати користувачам фішингові сторінки, інжектити шкідливе ПЗ або збирати облікові дані та іншу конфіденційну інформацію. У корпоративних мережах така підміна загрожує витоком критичних даних, компрометацією доступів і масштабними інцидентами безпеки.

Слід відзначити, що підміна DHCP-сервера не завжди має на меті навмисну атаку — часто це наслідок людської необережності або помилки в налаштуванні. Типова ситуація — співробітник чи відвідувач підключає до мережі домашній (SOHO) роутер із ввімкненим DHCP і під'єднує його LAN-портом. Після цього частина клієнтів «підчеплює» адреси від цього пристрою і вже отримує невірні маршрути, DNS або недійсні оренди (короткі lease), що призводить до помітних збоїв: від значного падіння швидкості до повної втрати доступу до локальних і зовнішніх сервісів.

Інші наслідки підміни DHCP можуть включати: конфлікти IP-адрес у мережі, часті перезапити DHCP (деякі клієнти постійно оновлюють оренду), фрагментацію мережевих шляхів і підвищене навантаження на інфраструктуру. Для адміністратора мережі характерні ознаки проблеми: клієнти з «чужими» шлюзами або DNS, несподівані зміни в таблицях маршрутизації, а також збільшення кількості DNS-помилок і інцидентів доступності.

Щоб мінімізувати ризик як навмисної підміни, так і випадкових проблем, зазвичай застосовують профілактичні заходи: жорстку сегментацію мережі, обмеження доступу до uplink-портів, увімкнення DHCP-snooping і пов'язаних функцій (Dynamic ARP Inspection, IP Source Guard) на комутаторах, блокування неавторизованих пристроїв та політики 802.1X для контролю підключення. Навіть прості правила — відокремлені гостьові VLAN і заборона підключення невідомих роутерів — значно знижують ймовірність виникнення подібних інцидентів.

Методи захисту

Найпростіший спосіб захисту від подібних атак — увімкнути на всіх комутаторах функцію DHCP snooping і правильно класифікувати порти. Потрібно виділити два типи інтерфейсів:

• **Довірені (trusted)** — порти, до яких підключено легітимний DHCP-сервер або uplink до іншого комутатора/маршрутизатора.

• **Недовірені (untrusted)** — порти для кінцевих пристроїв (робочі станції, гостьові ПК, IoT), на яких не повинно бути DHCP-серверів і через які найчастіше з'являються фальшиві сервери.

З увімкненим DHCP snooping комутатор моніторить DHCP-пакети (DHCPOFFER, DHCPACK) і дозволяє їх проходження тільки з довірених портів, а відповіді з недовірених інтерфейсів блокує. Запити клієнтів (DHCPDISCOVER, DHCPREQUEST) усе ще поширюються, але відповіді серверів від незареєстрованих портів не доходять до клієнтів. Це ефективно перешкоджає підміні DHCP-серверів у локальній мережі.

Практично це також дозволяє формувати таблицю прив'язок (IP ↔ MAC ↔ порт ↔ VLAN), яку можна використати для додаткових захисних механізмів (Dynamic ARP Inspection, IP Source Guard) і для виявлення аномалій. Під час налаштування бажано мінімізувати кількість trusted-портів, документувати їх і комбінувати DHCP snooping із сегментацією мережі, 802.1X-автентифікацією та моніторингом подій — це значно підвищить стійкість мережі до Rogue DHCP та суміжних загроз.

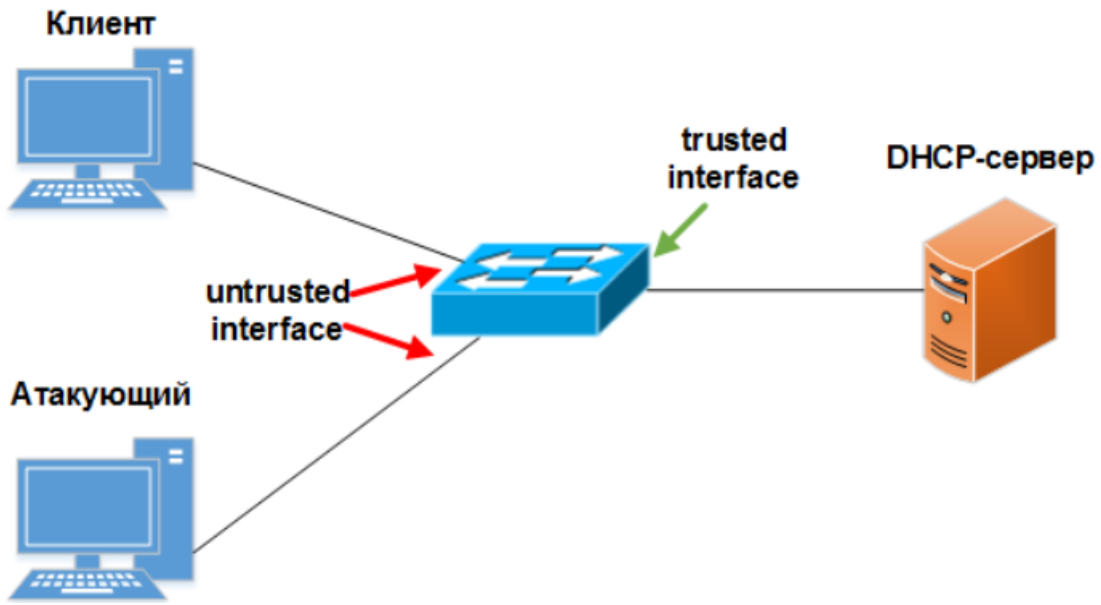


Рис. 3.2 Перенаправлення discover і request лише на довірені порти

Щоб правильно налаштувати механізм DHCP snooping на комутаторі, адміністратору необхідно виконати кілька послідовних кроків. Ця функція дозволяє забезпечити контроль за тим, які DHCP-сервери мають право обслуговувати клієнтів у мережі, і таким чином запобігає роботі несанкціонованих чи шкідливих DHCP-серверів.

1. **Активувати DHCP snooping на комутаторі.**
   Для початку потрібно увімкнути глобальний режим перевірки DHCP-повідомлень:
2. SW(config)# ip dhcp snooping

Це дає змогу комутатору почати відстежувати всі DHCP-пакети, що проходять через його інтерфейси.

3. **Задати VLAN, у яких буде контролюватися DHCP-трафік.**
   Як правило, у корпоративних мережах використовується кілька VLAN, але адміністратор може вибірково вказати ті з них, де найбільш критичним є захист від підміни DHCP-сервера:
4. SW(config)# ip dhcp snooping vlan <номер VLAN>

Таким чином, snooping буде працювати тільки в заданих сегментах, що дозволяє уникнути зайвого навантаження на обладнання.

5. **Позначити довірені інтерфейси.**
   На комутаторі потрібно вказати порти, через які дійсно підключені легітимні DHCP-сервери або uplink-лінії до інших комутаторів. Ці порти отримують статус **trusted**:
6. SW(config-if)# ip dhcp snooping trust

Усі інші порти за замовчуванням вважаються untrusted — тобто будь-які DHCP-відповіді (DHCPOFFER, DHCPACK), що надходитимуть із цих портів, комутатор відкидатиме. Це унеможливлює видачу адрес клієнтам від сторонніх пристроїв.

3.2 Атаки DHCP starvation

DHCP starvation — суть атаки й сценарії розвитку

DHCP starvation — це атака, спрямована на доведення до вичерпання пулу IP-адрес, який виділяє DHCP-сервер для певного VLAN/підмережі. DHCP-пул обмежений — наприклад, при масці 255.255.255.0 це 253 адреси користувачів — і за нормальних умов цього достатньо. Проте зловмисник може програмно (або з використанням простого інструментарію) багаторазово запитувати адреси, підмінюючи при цьому MAC-адресу, що дозволяє йому «забрати» велику кількість оренд (lease) і зробити пул порожнім.

Типовий сценарій дій зловмисника (сильно спрощено):

1. Зловмисник генерує DHCPDISCOVER/DHCPREQUEST з певною MAC-адресою → сервер видає IP (lease).
2. Потім атакуючий змінює MAC (спуфінг) і повторює запит під іншим MAC → отримує ще одну адресу.
3. Цикл повторюється, поки пул не буде вичерпано або поки адміністрація мережі не встигне відреагувати.

Наслідки такої активності бувають двоякі:

- **Відмова в обслуговуванні (DoS) для легітимних клієнтів** — нові пристрої не можуть отримати IP, відповідно втрачають доступ до мережі та сервісів. Це може паралізувати роботу відділу або сегмента мережі.
- **Комбінована атака з підміною DHCP (Rogue DHCP)** — після того, як легітимний сервер більше не має вільних адрес, підключений фейковий DHCP-сервер може почати обслуговувати запити клієнтів, видаючи некоректні шлюзи та DNS, реалізуючи MitM-сценарії.

Крім навмисних атак, подібна ситуація може виникнути випадково — наприклад, коли працівник підключає домашній SOHO-роутер в корпоративну мережу з увімкненим DHCP, або через помилкову конфігурацію relay/кешуючого пристрою.

Чому атака працює і де вона найбільш небезпечна

Причини ефективності атаки:

- DHCP — протокол без аутентифікації за контрактом (Trust model), тож сервери відповідають на широкомовні запити.

- Комутатори за замовчуванням дозволяють будь-якому кінцевому порту генерувати DHCP-трафік без обмежень.
- Легітимні мережі часто мають великі, «відкриті» VLAN для гостьових або IoT-пристроїв, де заходи контролю мінімальні.

Найбільше ризику піддаються відкриті офісні VLAN, мережі гостів, IoT-сегменти та будь-які середовища з необмеженим доступом до фізичних портів. У хмарних середовищах і дата-центрах такий вектор менш поширений (бо там пули та механізми видачі IP централізовані), але у поєднанні зі слабкими політиками віртуальної мережі атака може створити масштабні проблеми.

Методи захисту — технічні заходи та практики

Захист має бути багаторівневим: апаратні обмеження на комутаторах, політики доступу, моніторинг і реагування.

1) Port-security (обмеження MAC на порту)

Ідея: заборонити/обмежити кількість різних MAC-адрес, які може «вивчити» порт. Це зупиняє пристрій, який намагається мімікрувати під багато клієнтів, — комутатор просто не дозволить йому декорувати більше допустимого числа MAC.

Приклад базової конфігурації (Cisco-подібний синтаксис):

```
!-- Переключити порт в access режим
SW(config-if)# switchport mode access
SW(config-if)# switchport access vlan 10

!-- Увімкнути port-security
SW(config-if)# switchport port-security

!-- Обмежити максимум MAC-адрес (наприклад, 1 або 2 для робочої станції)
SW(config-if)# switchport port-security maximum 1

!-- Встановити sticky learning (адреси вивчаються і зберігаються в конфіг)
SW(config-if)# switchport port-security mac-address sticky
```

!-- Дія при порушенні

SW(config-if)# switchport port-security violation restrict

Режими реакції:

- protect — пакети з «нових» MAC відкидаються без логів;
- restrict — пакети відкидаються + запис у лог/SNMP;
- shutdown — порт переводиться в errdisable (блокування) і потребує відновлення (ручного або автоматичного).

Рекомендації: для кінцевих портів встановлюйте maximum 1 або 2 (для ноутбука + телефон через міст), використовуйте sticky для зручності, та restrict як баланс між безпекою і керованістю.

2) DHCP snooping

Ідея: дозволити відповіді DHCP (DHCPOFFER, DHCPACK) тільки з портів, які ви явно позначили як *trusted* (uplink або порт до DHCP-реле/серверу). Всі кінцеві порти — untrusted, і їхні DHCPOFFER відкидаються.

Базові команди:

!-- Глобально ввімкнути DHCP snooping

SW(config)# ip dhcp snooping

!-- Вказати VLANи, для яких працює snooping

SW(config)# ip dhcp snooping vlan 10,20

!-- Позначити uplink/серверний інтерфейс як trusted

SW(config-if)# ip dhcp snooping trust

Плюси: блокує небажані DHCPOFFER, дозволяє будувати binding-таблиці (IP↔MAC↔порт), які використовуються іншими захистами.

3) Rate limiting DHCP (snooping limit rate)

Обмежте кількість DHCP-пакетів на порті, щоб ускладнити швидку генерацію маси запитів:

SW(config-if)# ip dhcp snooping limit rate 10

10 pps або 50–100 pps — залежно від типу кінцевого обладнання. Рекомендації: для робочих станцій 10–20 pps достатньо; для гостьових/IoT — можна ставити нижче. При перевищенні — порт може перейти в errdisable.

4) Dynamic ARP Inspection (DAI) і IP Source Guard

DAI перевіряє ARP-запити/відповіді, зіставляючи їх з DHCP binding. Це зупиняє ARP-спуфінг і додатково перешкоджає MitM після підміни DHCP. **IP Source Guard** блокує пакети з IP, які не відповідають записам binding таблиці.

Команди (ідея):

!-- Увімкнути DAI для VLAN

SW(config)# ip arp inspection vlan 10

!-- Увімкнути IP source guard

SW(config-if)# ip verify source

5) Аутентифікація на доступі — 802.1X

Вимагаючи автентифікації на порті (EAP), ви практично унеможливлюєте підключення випадкових пристроїв, поки вони не пройдуть авторизацію. У парі з port-security і DHCP snooping значно підвищує захищеність.

6) VLAN-сегментація і гостеві мережі

Виділіть гостьові і IoT-пристрої в окремі VLAN з жорстким набором правил, невеликими DHCP-пулами та обмеженим доступом до внутрішніх ресурсів. Це зменшує blast radius у випадку атаки.

7) Фізичний захист і політики підключення

- обмежуйте фізичний доступ до портів;
- забороняйте підключення несертифікованих свічів/роутерів;
- ведіть інвентар мережевих портів і машин;
- використовуйте локальний ACL/port isolation.

8) Моніторинг, логування, SIEM та автоматичне реагування

Налаштуйте логування подій, пов’язаних із DHCP (кількість запитів, відхилені DHCPOFFER, port-security violations). Інтегруйте з SIEM для кореляції подій (масові DHCPREQUEST з одного порту + port-security violation → сповіщення

і автоматична ізоляція). Налаштуйте автоматичний скрипт, який при спрацюванні errdisable буде інформувати інженера та/або тимчасово знімати порт з мережі.

9) Налаштування поведінки при перевищенні лімітів (errdisable recovery)

Якщо ви обираєте політику shutdown для port-security — додатково вирішіть поведінку відновлення (errdisable recovery) і механізми нотифікації, щоб не створити критичну операційну проблему в разі помилкового спрацьовування.

10) Централізовані DHCP-рішення і relay (архітектурна міра)

Використання централізованого DHCP-серверу (наприклад, у віртуалізованому розподіленому середовищі) і relay-агентів мінімізує потребу в локальних DHCP-серверах і зменшує кількість trusted-портів. Реле повинні бути надійно захищені й розміщені в контрольованих сегментах.

Приклади розширених конфігурацій (Cisco-подібні)

1. Глобально ввімкнути DHCP snooping і обрати VLAN:

```
SW(config)# ip dhcp snooping
SW(config)# ip dhcp snooping vlan 10
```

2. Налаштувати trusted порт (uplink до DHCP сервера):

```
SW(config-if)# interface GigabitEthernet1/0/1
SW(config-if)# ip dhcp snooping trust
SW(config-if)# exit
```

3. Налаштувати port-security для кінцевого порту:

```
SW(config-if)# interface GigabitEthernet1/0/10
SW(config-if)# switchport mode access
SW(config-if)# switchport access vlan 10
SW(config-if)# switchport port-security
SW(config-if)# switchport port-security maximum 1
SW(config-if)# switchport port-security violation restrict
SW(config-if)# switchport port-security mac-address sticky
```

4. Ліміт DHCP пакетів на порті:

```
SW(config-if)# ip dhcp snooping limit rate 15
```

5. Динамічний ARP і IP Source Guard:

```
SW(config)# ip arp inspection vlan 10
SW(config-if)# interface GigabitEthernet1/0/10
SW(config-if)# ip verify source vlan 10
```

Виявлення атаки та інструменти аналізу

Ключові сигнали для виявлення DHCP starvation:

- Різкий сплеск DHCPDISCOVER/DHCPREQUEST з одного або кількох портів;
- Збільшення кількості виданих оренд до межі пулу;
- Port-security violations (лог/SNMP);
- Часті перезапити DHCP lease;
- Аномальна кількість ARP записів, змін у binding таблиці.

Інструменти: NetFlow/sFlow, SPAN/mirror порти + аналізатор трафіку (Wireshark, Bro/Zeek), IDS/IPS, SIEM. Для автоматичного виявлення створюйте кореляції в SIEM: наприклад, DHCP_DISCOVER_RATE > threshold + port_security_violation → тригер.

Реагування на інцидент (проведення інцидент-response)

Якщо виявлено DHCP starvation/підміну:

1. **Ізоляція порту(ів):** тимчасово відключити/перевести в errdisable підозрілі порти;
2. **Перевірка binding таблиць:** подивитися, які MAC/IP зайняли пул;
3. **Виявлення джерела:** аналіз SPAN/mirror, NetFlow, физичний огляд порту;
4. **Відновлення сервісу:** при потребі виділити резервний пул або тимчасово розширити DHCP-діапазон (з обережністю);
5. **Патчі/оновлення:** якщо виявлено експлуатацію вразливості в обладнанні — оновити ПЗ;
6. **Документування і preventive:** змінити політики доступу, додати 802.1X, підвищити ліміти/правила на комутаторах; повідомити користувачів.

Операційні рекомендації.

- Потрібно увімкнути **DHCP snooping** в усіх продуктивних VLAN, де клієнти отримують IP автоматично.
- Потрібно застосовувати **port-security** на крайніх портах (максимум 1–2 MAC).
- Потрібно налаштовувати **rate limiting** для DHCP-пакетів на портах користувачів (10–50 pps).
- Потрібно комбінувати **snooping** з **DAI** і **IP Source Guard** — цей набір дозволяє закрити більшість векторів атак, пов’язаних з DHCP і ARP.
- Потрібно реалізовувати **802.1X** для критичних сегментів; для гостьових мереж передбачати окремі VLAN з обмеженим пулом.
- Потрібно документувати всі **trusted** порти та мінімізувати їхню кількість.
- Потрібно підключати **моніторинг подій** і автоматизоване реагування через **SIEM**.
- Потрібно тестувати налаштування в лабораторії перед розгортанням у продуктиві, щоб уникати необачних відключень сервісу.

3.3 Атаки CAM-table overflow

Щоб зрозуміти механіку атаки типу CAM-table overflow (MAC flooding) і як їй протидіяти, корисно розширити опис роботи комутатора, показати, чому саме переповнення таблиці MAC дає атакуючому «гостиний доступ» до трафіку, і перелічити практичні заходи захисту та виявлення інциденту.

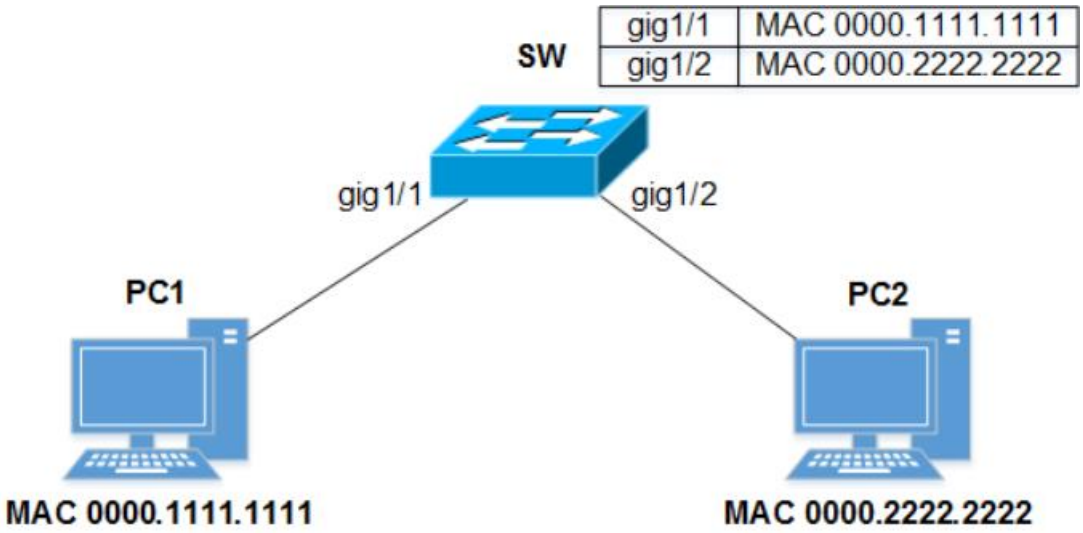


Рис. 3.3 Умовна початкова ситуація

Є комутатор SW з двома хостами: PC1 (MAC 0000.1111.1111, IP 10.0.0.1) і PC2 (MAC 0000.2222.2222, IP 10.0.0.2). Оскільки вони в одній підмережі, обмін пакетами йде через комутатор. Коли PC1 звертається до PC2 за IP, він надсилає ARP-запит (широкомовний): «Хто має IP 10.0.0.2 — відпишіть свій MAC». Комутатор ретранслює цей запит на інші порти, але водночас заносить у свою CAM-таблицю (MAC-таблицю) інформацію про порт, з якого надійшов ARP (тобто «MAC 0000.1111.1111 — порт X»). Коли PC2 відповідає, комутатор додає його запис. Після цього кадри між PC1 і PC2 вже пересилаються лише між відповідними портами — комутатор працює як інтелектуальний комутатор, а не як хаб.

Класичний сценарій атаки (кроки):

1. Атакуючий підключає пристрій до мережі або використовує скомпрометований вузол у VLAN.
2. За допомогою інструменту або скрипта (наприклад, macof з пакетом dsniff, або власні скрипти на Scapy) генерується маса кадрів з випадковими або послідовними MAC-адресами.
3. Комутатор записує ці фальшиві MAC-адреси в CAM-таблицю доти, доки вона не переповниться.
4. Після переповнення нові реальні адреси не потрапляють у таблицю — комутатор починає флудити трафік, і атакуючий має можливість перехоплювати пакети, читати незашифровані потоки або збирати метадані.

5. Цей ефект може розповсюджуватися і на сусідні комутатори через trunk-лінки, якщо вони передають фальшиві MAC-записи далі.

Наслідки атаки:

- перехоплення конфіденційних даних (якщо трафік не зашифрований);
- деградація продуктивності через додаткові широкомовні розсилки;
- підвищене навантаження CPU у комутаторах при обробці великої кількості кадрів;
- ускладнення роботи STP/маршрутизації при масових ARP/Spanning Tree-подіях;
- зручність для подальших атак (ARP-spoofing, session hijack, інжекція зловмисного трафіку).

Як виявити атаку (індикатори):

- раптовий приріст кількості записів у CAM/MAC-таблиці або швидке «старіння» записів;
- велика кількість неперервних ARP-запитів чи незвично високий рівень широкомовного трафіку;
- множинні port-security violations;
- збільшення числа пакетів на SPAN/mirror-порт або сигнали від систем IDS/IPS;
- користувачі скаржаться на зниження продуктивності або втрату сегмента мережі.

Практичні методи захисту (рекомендації, мультишарова стратегія)

1. **Port-security (обмеження MAC на порту).**
   — Увімкнути switchport port-security на всіх крайніх (access) портах.
   — Встановити maximum 1 (або 2, якщо очікується підключення ноут+телефон через Wi-Fi hotspot).
   — Використовувати sticky MAC-learning, щоб дозволити автоматичне «вивчення» першого законного MAC і збереження його в конфігурації.
   — Обрати режим реакції restrict або shutdown залежно від політики та готовності до відновлення порту.

Приклад (Cisco-подібно):

```
interface Gi1/0/10
 switchport mode access
 switchport access vlan 10
 switchport port-security
 switchport port-security maximum 1
 switchport port-security mac-address sticky
 switchport port-security violation restrict
```

2. **Використання DHCP snooping + Dynamic ARP Inspection + IP Source Guard.**

   — DHCP snooping створює binding-таблицю IP↔MAC↔порт, яка дозволяє DAI перевіряти ARP і IP Source Guard блокувати підміну IP на портах. Це додатково ускладнює масові маніпуляції з MAC/IP.

3. **802.1X / Network Access Control (NAC).**

   — Впровадження автентифікації портів (802.1X) з прив'язкою до облікових записів користувачів обмежує фізичний доступ і не дає зловмиснику вільно генерувати пакети з випадковими MAC.

4. **Мінімізація trusted-портів та контроль uplink.**

   — Мінімізація кількості trunk/uplink портів, які приймають MAC-записи; контролюйте, які VLAN транкуються; використовуйте VLAN pruning.

5. **Моніторинг і alerting.**

   — Налаштування збору метрик CAM-таблиць, port-security events, ARP-аналітику; інтегруйте з SIEM/IDS. Автоматичні алерти допоможуть швидко реагувати.

6. **Фізична безпека та політики підключення.**

   — Забезпечення контролю доступу до розподільних шаф; забороніть підключення несанкціонованих свічів/роутерів; ведіть інвентар портів.

7. **Шифрування трафіку на рівні застосунку/транспортного шару.**

— Навіть якщо кадри перехоплені, шифрування (TLS, SSH, IPsec transport) унеможливлює читання вмісту. Це критичний захід для захисту конфіденційних даних.

8. **Оновлення ПЗ і використання апаратних фільтрів.**

   — Використання останнбої версії прошивок, де виробник може реалізувати додаткові механізми захисту від MAC-flooding; розгляд модельні фільтри (hardware ACLs) для блокування аномалій.

Атака CAM-table overflow проста за ідеєю, але може мати серйозні наслідки в мережах зі слабкими політиками доступу. Надійний захист — це поєднання технічних обмежень на крайніх портах (port-security, storm-control), контролю автентифікації (802.1X), мережевого моніторингу і шифрування трафіку. Комбінований підхід значно знижує ймовірність успішної реалізації подібних атак і робить мережу більш стійкою до підслуховування та DoS-інцидентів.

4 ЗАХИСТ ВІРТУАЛЬНИХ МЕРЕЖ ВІД DDOS АТАК

DoS-атака (Denial of Service — відмова в обслуговуванні) полягає у масованій відправці на сервер жертви великої кількості пакетів, при цьому використовуються підроблені зворотні адреси. У результаті відбувається перевантаження каналу зв'язку, яким користується клієнт, або надмірне навантаження на ресурси самої системи, що призводить до збоїв у її роботі.

Щоб уникнути блокування за IP-адресою, зловмисники підміняють зворотні адреси у пакетах. Якщо ж атака здійснюється одночасно з багатьох пристроїв чи ботнетів, її вже класифікують як DDoS-атаку. Існує кілька найбільш поширених різновидів таких атак, які варто розглянути детальніше.

4.1 Атаки TCP SYN Flood

Мета атаки SYN Flood — викликати перевитрату обмежених ресурсів цільової системи та, як наслідок, призвести до відмови в обслуговуванні легітимних клієнтів. Механізм атаки базується на особливостях TCP-handshake: коли на сервер надходить TCP-SYN, операційна система резервує певні структури в пам'яті (внутрішні дескриптори з'єднань, стан сесії), поміщає запис у чергу очікування (backlog), генерує у відповідь SYN+ACK і чекає фінального ACK від клієнта. Ці дії вимагають CPU-часу й оперативної пам'яті для кожного напіввідкритого (half-open) з'єднання.

При інтенсивному потоці підроблених або неповних SYN-пакетів напіввідкриті записи накопичуються в таблицах з'єднань, черга backlog заповнюється, і сервер змушений відмовляти в обробці нових запитів — нові легітимні клієнти не можуть встановити з'єднання. Практичний поріг, який призводить до DoS, залежить від апаратних характеристик сервера, налаштувань TCP-стека і пропускної здатності мережі; як орієнтир часто наводять 100–500 тис. SYN/c як критичну зону для багатьох систем. Якщо нападник має потужний канал

(наприклад, гігабітний), він може генерувати мільйони пакетів на секунду і тим самим легко перевантажити навіть потужні інстанси.

Ще одна складність полягає в тому, що для ускладнення виявлення та блокування зловмисники часто спуфять (підміняють) IP-адреси джерел або використовують розподілені ботнети. У цьому випадку атака стає DDoS — вона походить одночасно з великої кількості різних адрес і стає значно складнішою для фільтрації простими ACL чи блокуванням по IP.

Щоб зменшити ризик і пом’якшити наслідки SYN-флуду, застосовують поєднання мережевих і системних заходів:

- **SYN cookies** у TCP-стеку — механізм, який дозволяє не виділяти ресурси для напіввідкритих з’єднань до отримання фінального ACK, шляхом кодування необхідного стану в значеннях пакета. Це ефективний спосіб захисту проти класичного SYN-flood.
- **Налаштування backlog і таймаутів** — збільшення розміру черги очікування і коригування таймаутів допомагає витримати короткочасні сплески, але не вирішує проблему при дуже інтенсивних хвилях.
- **Rate-limiting** на граничних інтерфейсах і балансувальниках — обмежує швидкість надходження SYN-пакетів від окремих потоків або сегментів мережі.
- **SYN-proxy / TCP-proxy** — проміжні проксі вирішують початковий handshake з клієнтом і лише після підтвердження запускають з’єднання до реального сервера, тим самим захищаючи бекенд від напіввідкритих сесій.
- **Архітектурні рішення** — anycast, CDN, розподілені балансувальники і DDoS-scrubbing сервіси провайдерів, що дозволяють рознести навантаження і очищувати трафік.

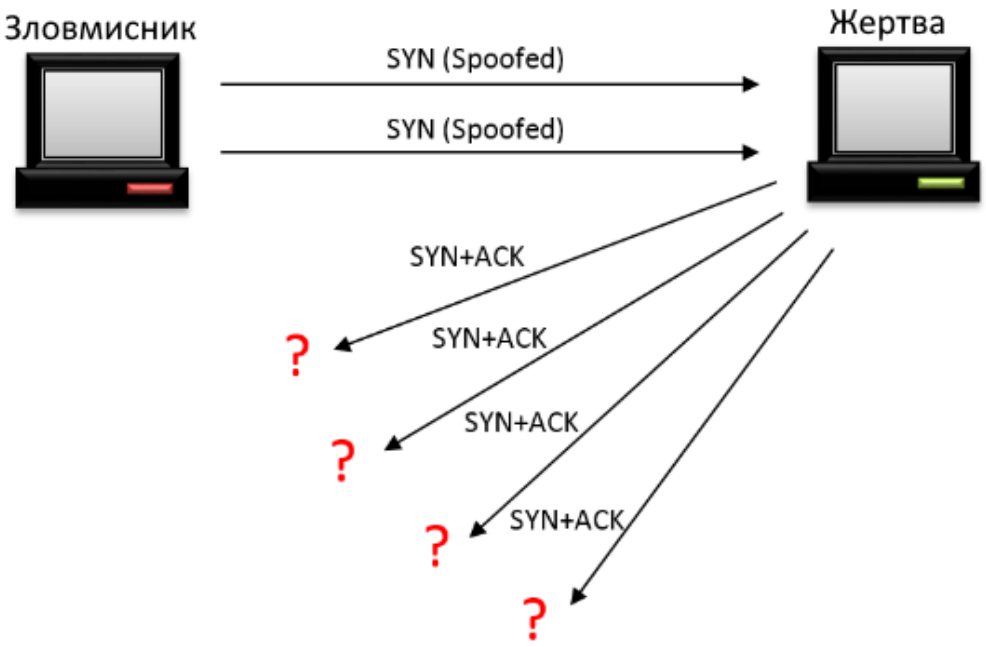


Рис. 4.1 Атака SYN Flood

Захист від SYN-флуду може також забезпечуватися за допомогою DPI-систем, які аналізують і фільтрують трафік у реальному часі на глибшому, сесійно-орієнтованому рівні. Замість того, щоб пропускати ці запити до реального веб-сайту, DPI-система сама відповідає на них і виконує перевірку клієнта. TCP-сесія з реальним сервером ініціюється тільки після того, як клієнт успішно пройшов перевірку (тобто підтвердив запит), — таким чином ресурси захищеного сервера не витрачаються на напіввідкриті з’єднання.

Такий підхід дає кілька практичних переваг: раннє виявлення аномалій у поведінці з’єднань, зниження навантаження на кінцеві сервіси, можливість застосувати додаткові перевірки (challenge–response, геолокаційні фільтри, репутаційні бази) і централізоване коригування політик. Водночас важливо розуміти компроміси: DPI-системи самі потребують ресурсів і правильної тонкої налаштування, щоб не блокувати легітимний трафік, тому їхнє впровадження й експлуатація мають бути частиною загальної DDoS-стратегії мережі.

У разі реальної атаки найбільш дієвими кроками є швидке включення захистів на рівні ОС (SYN-cookies), коригування rate-limits на крайових пристроях і негайна взаємодія з провайдером або активація scrubbing-послуг — ці заходи

зазвичай дають найшвидший і найпомітніший ефект у відновленні доступності сервісу.

4.2 Атаки за участі ботнету

Атаки за участі ботнету — один із найпотужніших і найпоширеніших способів організації DDoS-кампаній. Замість однієї машини зловмисник використовує тисячі або мільйони скомпрометованих пристроїв (ПК, сервери, IoT-пристрої), керованих централізовано або через розподілені канали. У результаті створюється масивний, синхронізований потік трафіку, який легко перевантажує мережеву інфраструктуру або ресурси сервера жертви.

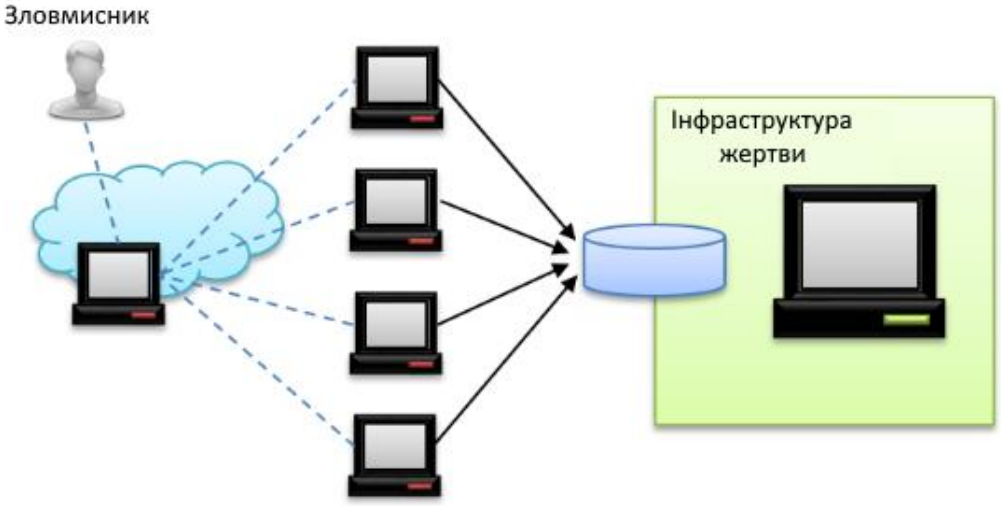


Рис. 4.2 Атаки за участі ботнету

Як це працює

- **Формування ботнету.** Зловмисник заражає пристрої шкідливим ПЗ, що переводить їх у стан «зомбі». Кожна така машина може виконувати команди нападника: генерувати трафік, встановлювати з’єднання, відправляти HTTP-запити, запускати SYN- або UDP-флуд тощо.
- **Канали управління (C2).** Центр керування ботнетом (C2) передає команди ботам через різні приховані канали: IRC-канали, P2P-мережі, зашифровані тунелі, DNS-канали, або навіть через пости в соціальних мережах (наприклад, Twitter), що ускладнює виявлення і блокування керування.

- **Атака.** У погоджений момент всі «зомбі» одночасно починають генерувати трафік на ціль — це можуть бути прості пакети (UDP/TCP-флуд), масивні HTTP-запити до важких ендпоінтів, або складніші сценарії, що імітують реальну поведінку користувачів, щоб обходити фільтри.

Чому це небезпечно

- **Масштаб і розподіленість.** Через велику кількість джерел фільтрація по IP часто неефективна.
- **Різноманітність векторів.** Атаки можуть бути багаторівневими: одночасно навантажувати мережеві інтерфейси, пули з'єднань і ресурсоємні обробки на прикладному рівні.
- **Залучення легітимних користувачів.** Деякі ботнет-атаки імітують легітимні запити, що ускладнює відокремлення «поганого» трафіку від «хорошого».

Приклади сценаріїв

- **Прямий флуд:** масова відправка UDP/TCP пакетів, що заповнюють канал або CPU цілі.
- **HTTP-DDoS:** сотні тисяч одночасних запитів до важких сторінок (login, search), що призводить до виснаження аплікаційних ресурсів.
- **Змішані атаки:** поєднання SYN-flood, DNS-ампліфікації і HTTP-флуду для максимального ефекту.

Виявлення та аналіз

- **Аномалійний моніторинг трафіку:** виявлення раптових сплесків pps/bitrate, незвичних патернів запитів (однакові user-agent, повторювані URI).
- **Агрегація логів і кореляція:** SIEM, NetFlow/sFlow/PCAP аналіз для знаходження розподілених джерел та загальних ознак.
- **Поведенчі алгоритми:** ML-моделі і евристики для виявлення бот-поведінки (нехарактерні часові шаблони, частота звернень, невідповідність геолокації/аутентифікації).

Методи захисту та пом'якшення

- **Rate-limiting і throttling:** обмеження швидкості запитів на прикордонних маршрутизаторах і балансувальниках.
- **Challenge–response (CAPTCHA, JS-challenge):** змусити клієнта виконати додаткову дію, що складна для ботів.
- **Anycast і розподіл навантаження:** рознести трафік по багатьох дата-центрах, щоб зменшити тиск на одну точку.
- **DDoS-scrubbing/cleaning-services:** перенаправлення трафіку через провайдера чи спеціалізований сервіс, який фільтрує і повертає «чистий» трафік.
- **WAF і DPI:** аплікаційні фаєрволи та DPI-системи, що аналізують сесії і блокують патерни атак.
- **Блокування і фільтрація на провайдера:** координація з upstream-операторами для відсікання атак на мережевому рівні (blackholing, remote triggered black hole, RTBH).
- **Автоматичне скалювання інфраструктури:** хмарні рішення, які дозволяють миттєво масштабувати ресурси, щоб витримати піки навантаження.

Операційні рекомендації

- Налаштувати моніторинг аномалій і швидкі алерти при перевищенні порогів pps/bitrate.
- Мати план реагування на DDoS: контакти провайдерів, списки IP-блоків, сценарії перенаправлення на scrubbing.
- Використовувати багаторівневу систему захисту: мережеві фільтри → балансувальники/Anycast → WAF → scrubbing.
- Регулярно сканувати та оновлювати пристрої в мережі, щоб зменшити ризик створення ботнет-узлів в своїй інфраструктурі.

4.3 Інші види атак

Smurf-атаки

Суть атаки. При Smurf-атакі зловмисник формує підроблений ICMP Echo-запит (ping), у полі «джерело» вказує IP-адресу жертви, а сам пакет спрямовує на адресу широкомовної розсилки (broadcast) певної мережі або підмережі. Оскільки пакет адресований широкомовно, усі хости в цій підмережі отримують запит і відповідають ICMP Echo Reply — але відповідь надсилається вже на підставлену адресу жертви. Таким чином один вихідний запит множиться на число вузлів у «підсилювальній» мережі, що дає ефект ампліфікації: відправивши один пакет у мережу з 100 хостів, атакувальник може викликати приблизно 100 відгуків на адресу жертви.

Небезпека. Smurf-атака навантажує канал жертви і її мережеве обладнання, призводить до значного зростання вхідного трафіку, який може заповнити пропускну здатність або призвести до виснаження ресурсів (CPU) на приймаючих пристроях. Особливо ефективні такі атаки проти мереж із великою кількістю незахищених або неправильно налаштованих хостів.

Захист. Щоб уникнути підсилювального ефекту, фахівці рекомендують:

- Заабілітувати (заборонити) пересилання прямих широкомовних ICMP-echo запитів на граничних маршрутизаторах і фільтрувати такі пакети на межі мережі.
- Налаштувати кінцеві хости/сервери та маршрутизатори так, щоб вони не відповідали на широкомовні ICMP-запити (включити режим «тихого» відкидання broadcast-echo).
- Забезпечити, щоб провайдери та адміністратори мереж не допускали маршрутизаторів у режимі «directed broadcast», якщо це не необхідно для специфічних сервісів.

DNS-атаки з підсиленням (DNS amplification)

Суть атаки. DNS-ампліфікація — це тип DDoS, у якому зловмисник надсилає невеликі DNS-запити до відкритих рекурсивних резолверів, підмінюючи

в полі «джерело» IP жертви. Резолвер, виконуючи запит, повертає великий обсяг відповіді (наприклад, через використання DNS-SEC або специфічних типів записів), і вся ця велика відповідь надходить на адресу жертви. Співвідношення розміру відповіді до запиту може бути дуже високим, тому загальний трафік, спрямований на жертву, збільшується в десятки або сотні разів.

Історичний приклад. У лютому 2007 року відбулися масовані напади на кореневі DNS-сервери, що продемонструвало вразливість інфраструктури DNS до таких підсилювальних методів і змусило спільноту впроваджувати практики захисту.

Захист і практики:

- Закривати або обмежувати доступ до рекурсивних резолверів тільки для довірених клієнтів; забороняти open-resolver.
- Впроваджувати rate-limiting на DNS-запити і відповіді, фільтрувати підозрілу активність на межі мережі.
- Використовувати DNS-рекурсери з можливістю фільтрації аномалій; співпрацювати з провайдерами для блокування підсилювальних джерел.
- Слідкувати за оновленнями рекомендацій RIPE/ICANN/Cisco та впроваджувати best practices щодо налаштувань DNS-інфраструктури.

TCP Reset (RST)-атаки

Суть атаки. У TCP-протоколі пакет із прапорцем **RST** призначений для негайного скидання з’єднання. Якщо зловмисник підмінює або інжектить RST-пакети з правильними значеннями (IP, порти, sequence/ack), ці пакети можуть чинити вигляд валідних і змушувати кінці сесій миттєво розривати з’єднання. В результаті легітимні з’єднання постійно обриваються — користувачі не можуть завершити трансфер, а служби втрачають доступність.

Захист. Стримування RST-інжекцій потребує більш глибокого аналізу контексту пакетів:

- Використовувати DPI/NGFW рішення, які відстежують коректність sequence/ack і виявляють підроблені RST-пакети;

- Вводити механізми повторної валідації сесії на рівні додатка або TLS-шари (якщо застосовно);
- Використовувати захищені транспортні канали (TLS), де розрив сесії не відбувається лише через прийом RST-пакета на мережевому рівні.

Загальний контекст — роль операторів і сервісів захисту

Сьогодні багато атак мають за мету не просто компрометацію окремих машин, а організацію масштабних DDoS-кампаній або перешкоджання доступу до сервісів. У відповідь оператори зв’язку, хостинг-компанії й спеціалізовані провайдери захисту пропонують комплексні інструменти:

- **Реальний моніторинг трафіку** з виявленням аномалій і тривожних сплесків (pps/bitrate).
- **DDoS-scrubbing / cleaning-services** — перенаправлення трафіку через фільтраційні центри, видалення шкідливих потоків і повернення «чистого» трафіку.
- **Anycast / CDN / розподіл навантаження** — географічний розподіл сервісів для зменшення впливу на окрему точку.
- **Carrier-grade NAT (CGNAT)** — допомагає приховати внутрішні пристрої від прямого доступу з інтернету, що ускладнює використання їх як цілей або джерел атак.
- **Інтелектуальні і самообучувані системи** аналізу поведінки, які корелюють події, виявляють нові патерни атак і автоматично застосовують політики захисту.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи на тему «Методи захисту віртуальних мереж від атак» було проведено комплексне дослідження сучасних загроз, що виникають у процесі функціонування віртуалізованих середовищ, та визначено найбільш ефективні методи протидії цим загрозам.

Аналіз показав, що віртуальні мережі, завдяки своїй гнучкості, масштабованості та широким можливостям інтеграції з хмарними та програмно-орієнтованими рішеннями, одночасно стають і привабливою ціллю для зловмисників. Класичні вектори атак, зокрема DoS/DDoS, TCP SYN Flood, Smurf-атаки, DNS-ампліфікація, MAC-spoofing, VLAN-hopping, CAM table overflow, підміна DHCP-серверів та DHCP starvation, набувають нового звучання у контексті віртуальних середовищ, оскільки вони можуть паралізувати роботу цілих сегментів або навіть критичних сервісів.

Особливу увагу приділено проблемам, пов’язаним із маніпуляціями над алгоритмами машинного навчання, які все частіше застосовуються у захисних механізмах віртуальних мереж. Дослідження продемонструвало, що adversarial machine learning може стати інструментом атак, коли підготовлені дані здатні знизити ефективність систем детекції чи навіть «переобучити» модель на користь зловмисника. Це вимагає поєднання класичних методів захисту (фільтрації, шифрування, контроль доступу) із новітніми підходами на основі поведінкового аналізу та генеративно-змагальних мереж.

У роботі розглянуто й апаратні аспекти безпеки: захист пам’яті (PMP у RISC-V, Paging Protection у x86), контроль привілеїв (рівні доступу у RISC-V, x86 та ARM із TrustZone), а також методи запобігання спекулятивним атакам типу Spectre та Meltdown. Ці механізми мають безпосередній вплив на безпеку віртуалізованих систем, оскільки дозволяють локалізувати виконання потенційно небезпечних процесів і зменшити ризик несанкціонованого доступу до ресурсів.

Запропоновані заходи захисту віртуальних мереж передбачають використання DHCP snooping, Dynamic ARP Inspection, IP Source Guard та port-security для

нейтралізації атак другого рівня, застосування шифрування трафіку та тунелювання для протидії перехопленню даних, впровадження rate limiting і поведінкового аналізу трафіку для мінімізації наслідків DoS/DDoS-атак, а також використання DPI-систем у поєднанні з SIEM-рішеннями для моніторингу та автоматизованого реагування. Важливу роль відіграє також застосування апаратних функцій безпеки сучасних процесорних архітектур, що дозволяють контролювати доступ до пам’яті й ізолювати потенційно небезпечні процеси у віртуалізованих середовищах.

Узагальнюючи результати, можна стверджувати, що ефективний захист віртуальних мереж вимагає багаторівневого підходу, де класичні методи мережевої безпеки інтегруються з новими технологіями інтелектуального аналізу трафіку та апаратними механізмами захисту. Важливим є не лише блокування відомих атак, але й підготовленість до появи нових, що враховує тенденцію до зростання складності загроз і поширення штучного інтелекту як інструменту як захисту, так і атак.

Таким чином, результати дослідження підтверджують, що майбутнє захисту віртуальних мереж полягає у поєднанні адаптивних, інтелектуальних та апаратно-програмних методів, що здатні забезпечити стабільність і безпеку функціонування критичних комунікаційних інфраструктур.

ПЕРЕЛІК ПОСИЛАНЬ

1. Virtualization Security: Protecting Virtualized Environments. Wiley, 2015. – захист віртуалізації: гіпервізори, ізоляція, контроль доступу.
2. Virtualization for Security. Hoopes, J. Elsevier, 2008. – використання віртуалізації для безпеки: sandbox, honeypot, forensic.
3. Guide to Security in SDN and NFV: Challenges, Opportunities, and Applications. Zhu, S.-Y., Scott-Hayward, S., Jacquin, L., Hill, R. Routledge, 2022. – безпека SDN/NFV: архітектура, політики, захист служб.
4. Hardware and Software Support for Virtualization. Bugnion, E., et al. Morgan & Claypool, 2011. – фізичні та програмні основи віртуалізації, уразливості.
5. SDN and NFV: A New Dimension to Virtualization. Gupta, B. B., et al. World Scientific, 2024. – синергія SDN та NFV у віртуальних мережах і методи безпеки.
6. Security and Privacy Preserving for IoT and 5G Networks: Techniques, Challenges, and New Directions. Gupta, B. B., et al. Springer, 2022. – методи захисту у віртуальних мережах IoT/5G.
7. Handbook of Computer Networks and Cyber Security: Principles and Paradigms. Springer, 2020. – загальні принципи безпеки мереж і віртуалізації.
8. Network Security Essentials: Applications and Standards. Stallings, W. Pearson, 2021. – криптографія, протоколи, стандарти захисту.
9. Cybersecurity – Attack and Defense Strategies. O'Reilly, 2021. – методи захисту віртуальних інфраструктур і сегментації.
10. Software-Defined Networking and Security: From Theory to Practice. CRC Press, 2023. – практичні аспекти безпеки в SDN.
11. Bays, L. R., et al. Virtual Network Security: Threats, Countermeasures, and Challenges. Journal of Internet Services and Applications, 2015. – загрози та контрзаходи у віртуальних мережах.
12. Alnaim, A. K., et al. Securing 5G Virtual Networks: A Critical Analysis of SDN, NFV, and Network Slicing. International Journal of Information Security, 2024. – аналіз безпеки SDN, NFV і сегментації.

13. Zohaib, S. M., et al. Zero Trust VPN (ZT-VPN): A Systematic Literature Review and Cybersecurity Framework. Information, 2024. – Zero-Trust VPN для захисту віртуальних каналів.
14. Domínguez-Dorado, M., et al. Detection and Mitigation of Security Threats Using SDN. Applied Sciences, 2023. – захист SDN: контролер, комунікації, атаки.
15. Doriguzzi-Corin, R., et al. Dynamic and Application-Aware Provisioning of Chained Virtual Security Network Functions. Computer Communications, 2019. – розгортання віртуальних функцій безпеки.
16. Zhang, P., et al. Security-Aware Virtual Network Embedding Algorithm Based on Reinforcement Learning. 2022. – алгоритм вбудовування віртуальних мереж із врахуванням безпеки.
17. He, J., et al. NetSecCC: A Scalable and Fault-Tolerant Architecture without Outsourcing Cloud Network Security. 2014. – архітектура захисту у хмарних і віртуальних мережах.
18. Virtual Private Networks: Fundamentals, Security Issues and Solutions. 2023. – VPN, уразливості та методи захисту.
19. Enhancing Security in SDN: Systematizing Attacks and Vulnerabilities. Computer Networks, 2024. – систематизація атак і вразливостей у SDN.
20. A Comprehensive Review on SDN and DDoS Attacks: Ecosystem, Taxonomy, Challenges. Computer Standards & Interfaces, 2025. – DDoS у SDN, методи протидії.
21. A Survey on Data Plane Security in SDN: Toward Adaptive Security. 2025. – захист data-plane та адаптивні рішення.
22. SDN as a Defence Mechanism: A Comprehensive Survey. International Journal of Information Security, 2024. – використання SDN як механізму захисту.
23. EnsureS: Secure Service Path Validation in SDN. Scientific Reports, 2023. – валідація сервісних шляхів, цілісність.
24. Q-MIND: Defeating Stealthy DoS Attacks in SDN with Machine Learning. 2019. – ML-підхід для захисту від прихованих DoS.

25. Unseen Attack Detection in SDN Using BERT-Based Models. 2024. – виявлення невідомих атак за допомогою LLM.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ

КВАЛІФІКАЦІЙНА РОБОТА

МЕТОДИ ЗАХИСТУ ВІРТУАЛЬНИХ МЕРЕЖ ВІД АТАК

ВИКОНАВ:
СТУДЕНТ ГРУПИ ТСДМ-62
ОЛЕГ ЛЕВЧИК

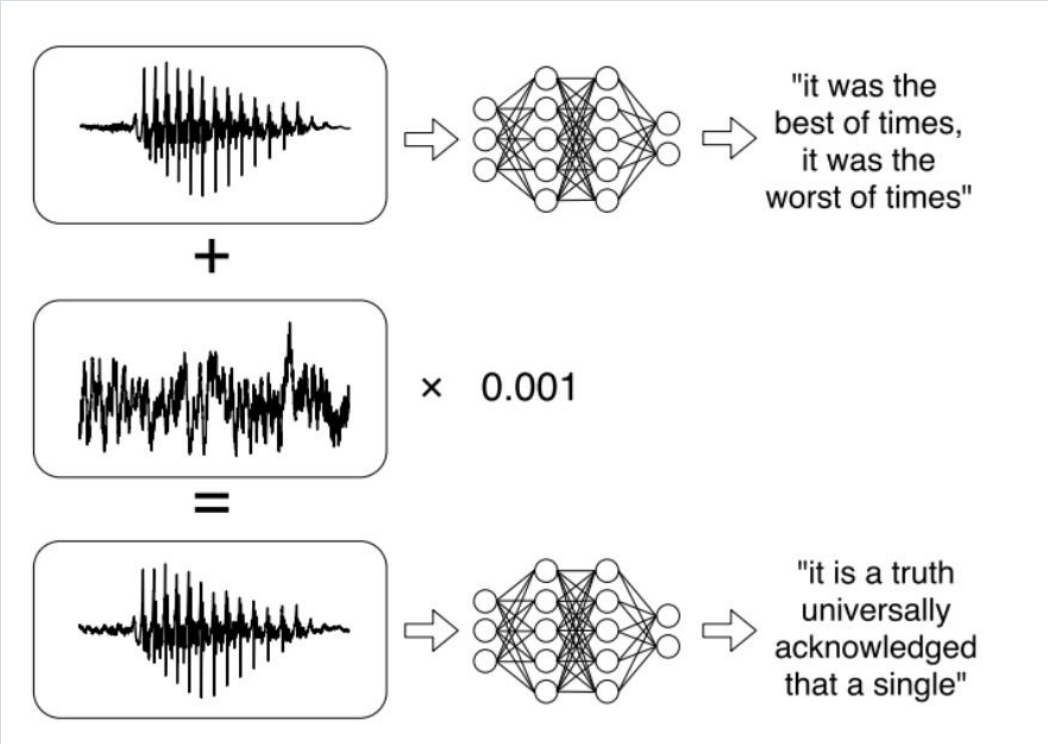
МЕТА РОБОТИ ТА АКТУАЛЬНІСТЬ

- Мета роботи – систематизувати і оцінити сучасні підходи до захисту віртуальних мереж, розробити практично застосовну багаторівневу архітектуру захисту та запропонувати методики виявлення і протидії як класичним мережевим атакам, так і новим викликам, з урахуванням апаратних можливостей сучасних процесорних архітектур.
- Об’єкт дослідження – віртуалізовані мережеві середовища та інфраструктури (включаючи віртуальні машини, віртуальні комутатори/маршрутизатори, SDN та хмарні платформи).
- Предмет дослідження – методи, алгоритми та технічні механізми виявлення, запобігання і мінімізації наслідків атак у віртуальних мережах (мережеві політики, DPI, SIEM, поведінковий аналіз, захист на апаратному рівні, заходи проти adversarial ML).

Актуальність:

- Тема роботи є актуальною, тому що стрімкий розвиток інформаційних технологій, хмарних обчислень і сервісів віртуалізації призвів до того, що віртуальні мережі стали невід’ємною складовою сучасних телекомунікаційних систем. Вони забезпечують масштабованість, гнучкість і оптимальне використання ресурсів, дозволяючи реалізовувати комплексні корпоративні й інфраструктурні рішення. Проте поряд із перевагами віртуальні середовища породжують нові виклики у сфері кібербезпеки, оскільки традиційні механізми захисту не завжди здатні ефективно протидіяти специфічним загрозам у таких умовах.

Використання методів машинного навчання в віртуальних мережах



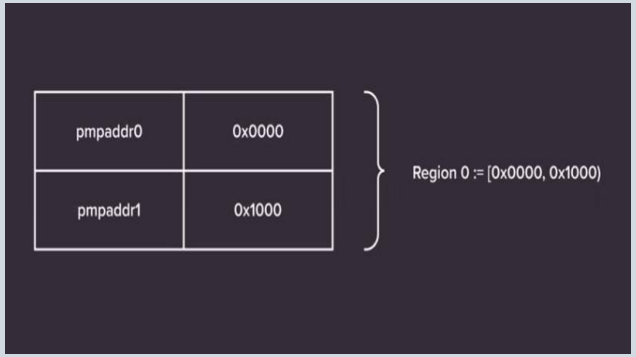
Приклад цілеспрямованої атаки на систему автоматичного розпізнавання мовлення (ASR)

У віртуальних мережах **машинне навчання (ML)** оптимізує маршрутизацію, балансування навантаження та виявлення аномалій, підвищуючи **надійність і ефективність**. Однак моделі ML можуть стати об’єктом атак: **отруєння даних**, фальсифікація телеметрії, маніпуляції контролерами SDN/NFV та спотворення класифікаторів трафіку. Ці дії поступово деградують продуктивність і обходять політики безпеки.

Захист включає **робастне навчання, adversarial training, моніторинг дрейфу, перевірку даних, ансамблі моделей, human-in-the-loop та аудит**. Приклад: атаки на системи розпізнавання мовлення (ASR) через невидимі адверсарні збурення, що змінюють транскрипцію.

ML у віртуальних мережах — потужний інструмент, але вимагає **комплексного підходу до безпеки**, враховуючи найгірші сценарії атак.

Захист пам’яті в архітектурах віртуальних мереж



Приклад запису довжини області, що захищається в регістр pmpaddr

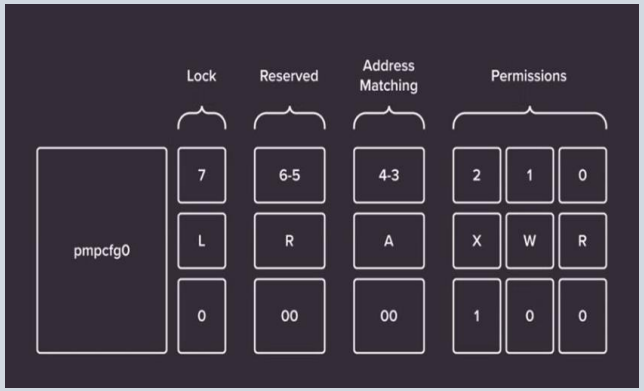
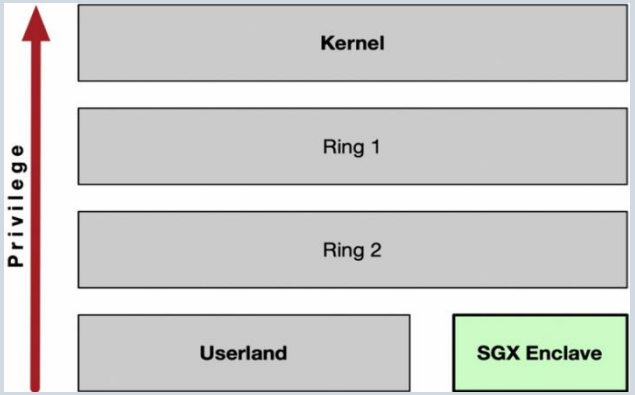
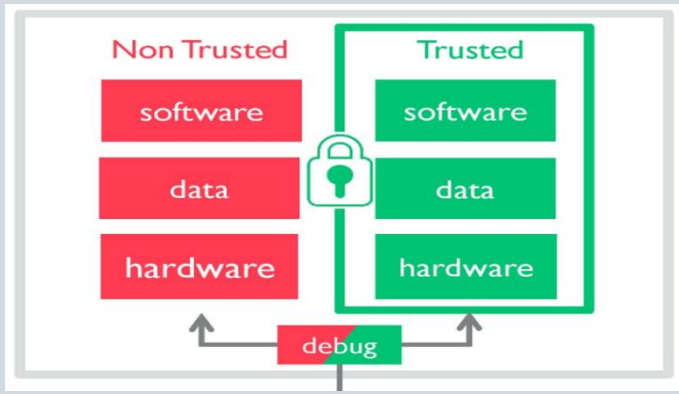


Схема запису байтів у регістр pmpcgf0



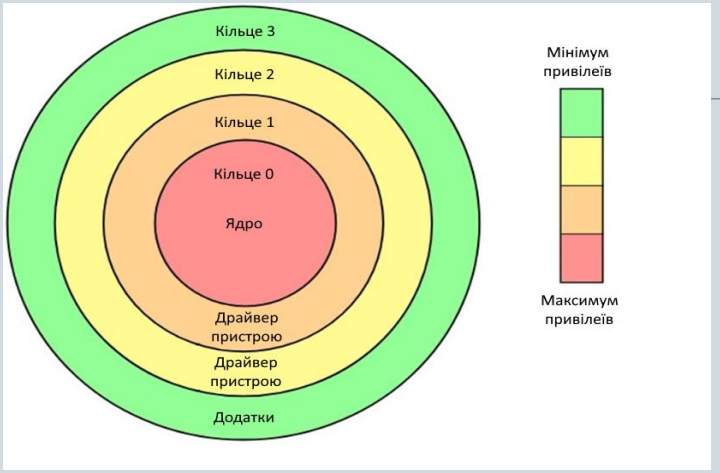
Робота анклаву Intel SGX



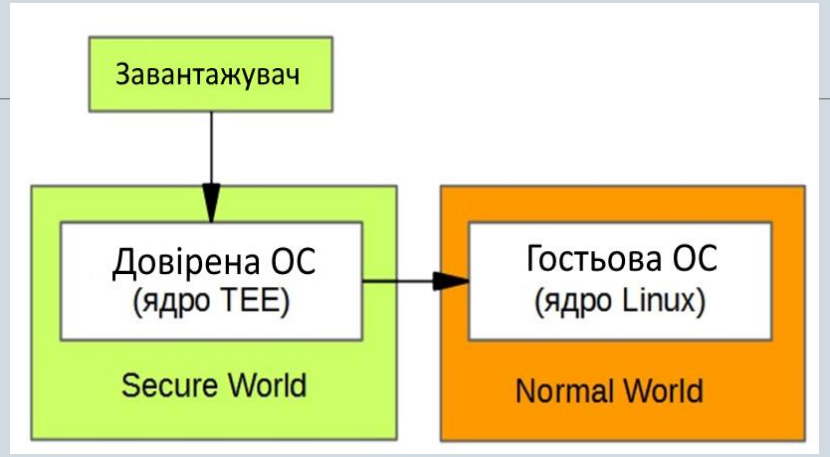
Приклад роботи TrustZone

У RISC-V захист пам’яті реалізується через PMP (Physical Memory Protection), який обмежує доступ до фізичних ділянок пам’яті за допомогою регістрів pmpaddr та pmpcfg (R/W/X, lock). PMP ефективний для вбудованих і IoT-систем, але обмежений кількістю регістрів. В архітектурі x86 застосовується Paging Protection із віртуальною пам’яттю, сторінками та кільцевими рівнями привілеїв, що підходить для складних ОС і серверів. Для захищених обчислень існують TEE: Intel SGX (анклави в x86) забезпечує апаратну ізоляцію, ARM TrustZone розділяє систему на Secure і Normal World, ізолюючи критичні дані та операції. Keystone Enclave на RISC-V відкритий і гнучкий для хмар та віртуальних мереж.

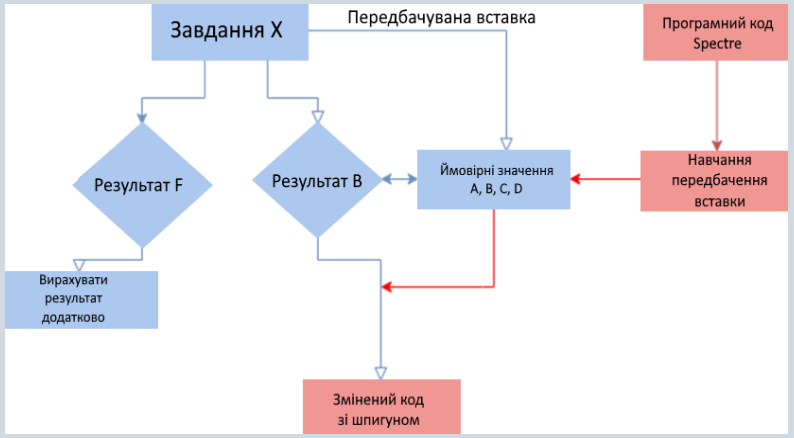
Контроль захисту в процесорах віртуальних мереж



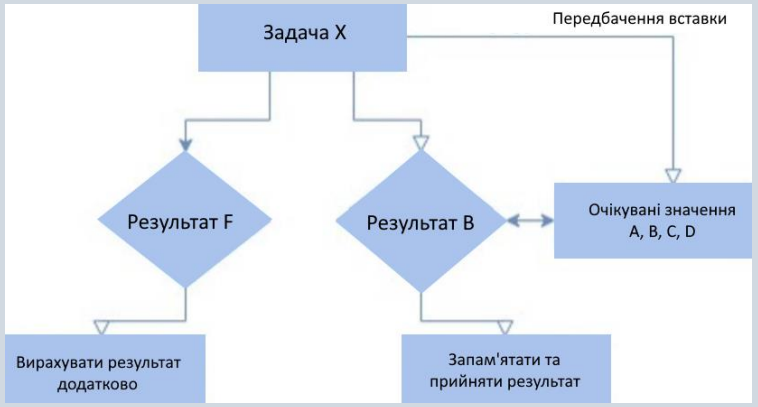
Рівні привілеїв x86



Рівні привілеїв в ARM



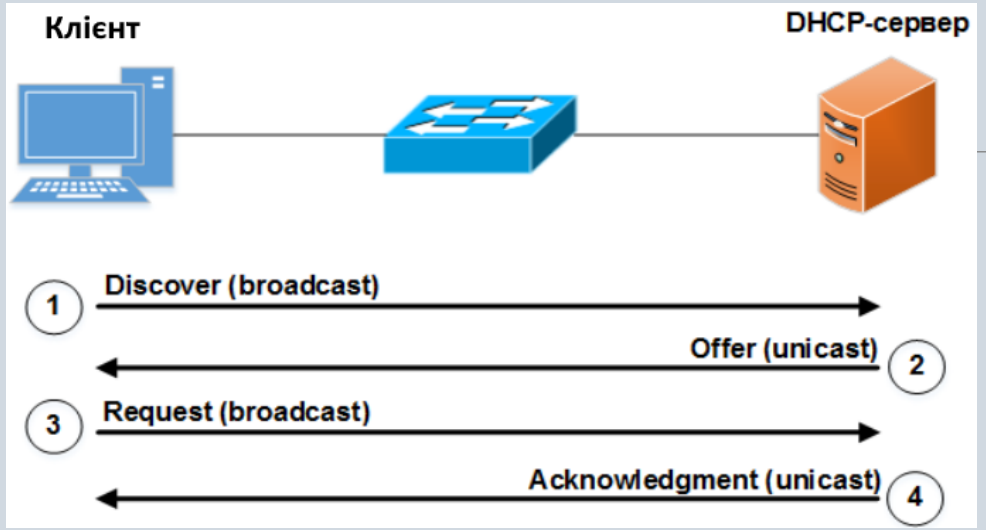
Приклад роботи Spectre-атаки



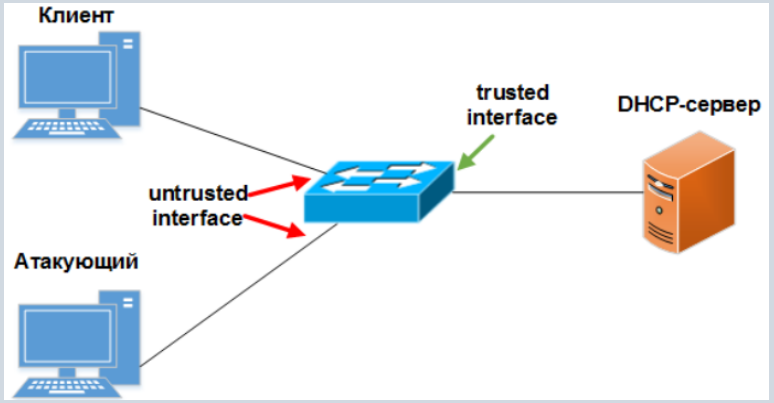
Приклад роботи Spectre-атаки

Контроль привілеїв у сучасних процесорах забезпечує безпеку та стабільність систем, ізолюючи користувацькі програми від ядра та апаратури. У **RISC-V** реалізовані три рівні: **M-mode** (апаратні ресурси), **S-mode** (ОС) та **U-mode** (користувач). У **x86** — чотири кільця (**Ring 0–3**), на практиці застосовуються лише ядро та користувацькі програми. **ARM TrustZone** розділяє систему на **Secure** та **Normal World**, забезпечуючи апаратну ізоляцію. Спекулятивне виконання, як у **Spectre** та **Meltdown**, створює загрози. RISC-V може обходитися без нього, ARM і x86 застосовують інструкції LFENCE, CSDB, SSBS для захисту. Механізми критичні у віртуалізації та хмарних середовищах.

Rogue DHCP Server і DHCP starvation



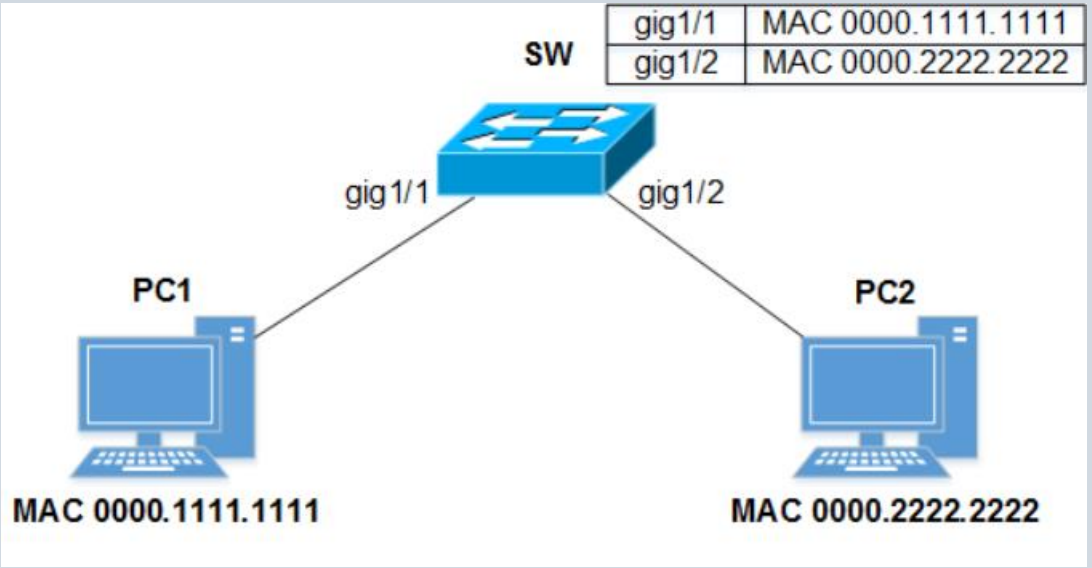
Підміна DHCP-сервера



Перенаправлення discover і request лише на довірені порти

Атаки **Rogue DHCP Server** і **DHCP starvation**. Rogue DHCP Server — несанкціонований сервер, що видає некоректні IP, шлюзи та DNS, дозволяючи MitM-атаки, фішинг і витік даних. DHCP starvation виснажує пул IP-адрес, блокуючи легітимних клієнтів і відкриваючи шлях для Rogue DHCP. Захист включає **DHCP Snooping**, **Dynamic ARP Inspection**, **IP Source Guard**, **port-security**, **rate limiting**, **802.1X**, VLAN-сегментацію та моніторинг подій через SIEM/IDS. Рекомендовано мінімізувати trusted-порти, контролювати доступ до фізичних портів і тестувати налаштування перед впровадженням у продуктивній мережі.

Атаки CAM-table overflow



Умовна початкова ситуація

Атака: переповнення CAM-таблиці комутатора фальшивими MAC, що дозволяє перехоплювати трафік та знижує продуктивність.

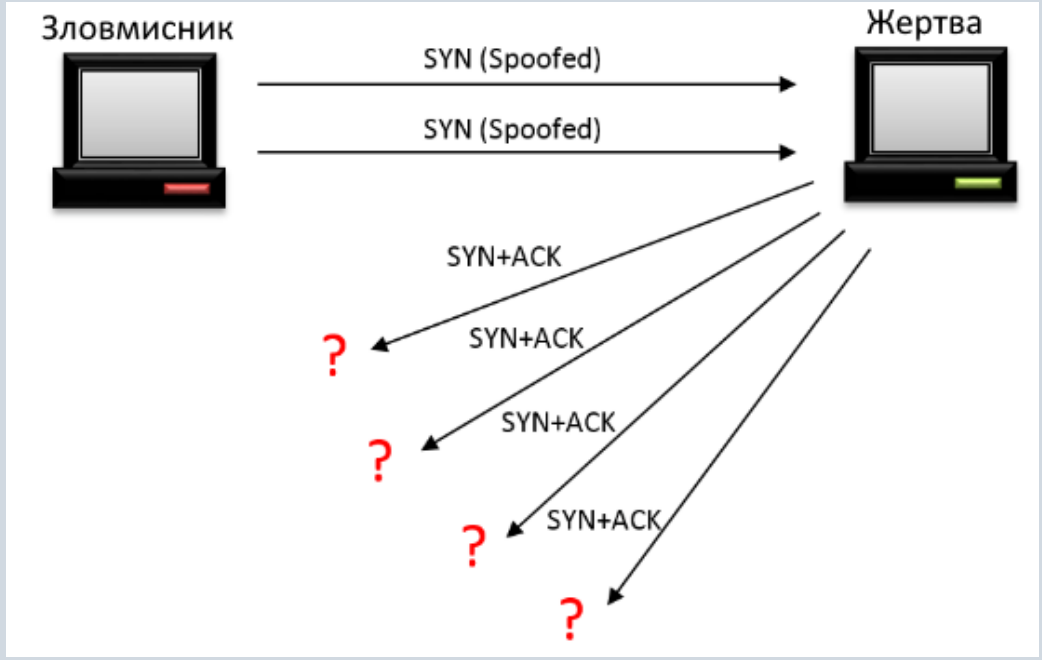
Виявлення: різке заповнення MAC-таблиці, масові ARP-запити, port-security violations, підозрілі пакети на SPAN/mirror-порту.

Захист:

- Port-security (обмеження MAC, sticky MAC, режим restrict/shutdown);
- DHCP snooping + Dynamic ARP Inspection + IP Source Guard;
- 802.1X/NAC для автентифікації портів;
- Контроль uplink, VLAN pruning;
- Моніторинг і алерти через SIEM/IDS;
- Шифрування трафіку (TLS/SSH/IPsec);
- Оновлення прошивок і апаратні фільтри.

Висновок: комбінований захист підвищує безпеку і стійкість мережі.

Атаки TCP SYN Flood



Атака SYN Flood

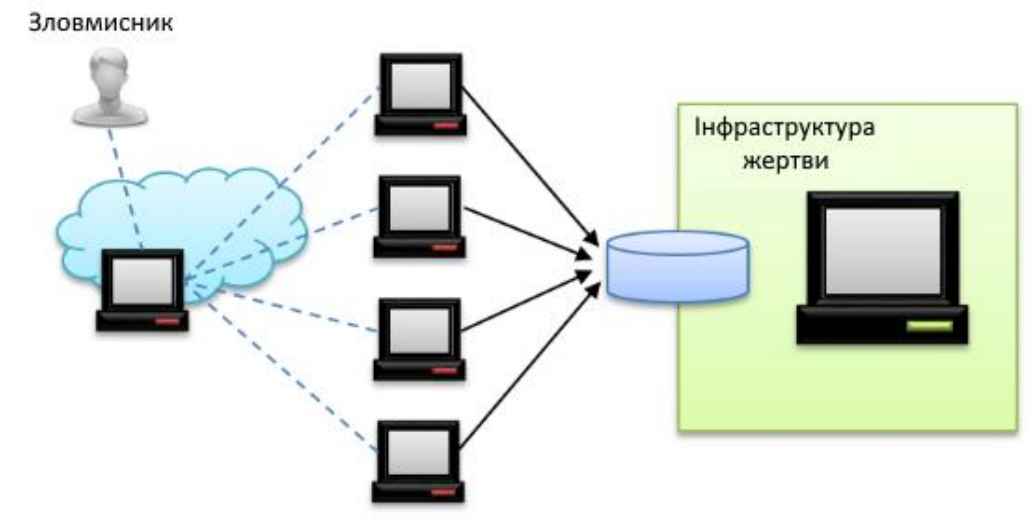
SYN Flood спрямований на виснаження ресурсів серверу через напіввідкриті TCP-з'єднання: при отриманні SYN сервер резервує ресурси, відправляє SYN+ACK і очікує фінального ACK; інтенсивний потік підроблених SYN швидко заповнює backlog і перешкоджає легітимним з'єднанням. Атаки часто ускладнюють спуфінгом джерел або розподіленими ботнетами, що перетворює їх у DDoS.

Захист і пом'якшення:

- SYN cookies — уникнення резервування стану до підтвердження;
- Налаштування backlog і таймаутів;
- Rate-limiting на граничних інтерфейсах;
- SYN-proxy / TCP-proxy для завершення handshake замість бекенду;
- Anycast/CDN і scrubbing-сервіси для розподілу/очищення трафіку;
- DPI/NGFW для сесійного аналізу і фільтрації.

При атаці першочергові дії — увімкнення SYN-cookies, коригування rate-limit і звернення до провайдера або активація scrubbing-послуг.

Атаки за участі ботнету та інші види атак



Атаки за участі ботнету

Ботнет-атаки

Ботнети використовують тисячі заражених пристроїв для масованого DDoS, перевантажуючи канали і сервери. Команди надсилаються через C2-канали (IRC, P2P, DNS, соцмережі). Захист: rate-limiting, CAPTCHA, Anycast, scrubbing, WAF/DPI, масштабування і координація з провайдерами.

Інші атаки

- **Smurf:** ICMP broadcast на IP жертви → перевантаження каналу. Захист: блокування широкомовних запитів.
- **DNS-ампліфікація:** великі відповіді на підроблені запити. Захист: закриття open-resolvers, rate-limiting.
- **TCP Reset:** підроблені RST-пакети обривають сесії. Захист: DPI, TLS, перевірка sequence/ack.

Загальний захист: scrubbing, Anycast/CDN, CGNAT, системи поведінкового аналізу для виявлення і блокування атак.

ВИСНОВКИ

- У результаті виконання кваліфікаційної роботи проаналізовано сучасні загрози для віртуальних мереж та визначено ефективні методи їх захисту. Показано, що віртуальні середовища є вразливими до класичних і нових типів атак, зокрема DDoS, VLAN-hopping та маніпуляцій із моделями машинного навчання.
- Запропоновано комплекс заходів, що поєднує класичні методи (фільтрація, шифрування, контроль доступу) з інтелектуальними підходами аналізу трафіку та апаратними механізмами безпеки.
- Зроблено висновок, що надійний захист віртуальних мереж забезпечується багаторівневою системою, яка інтегрує програмні, апаратні та інтелектуальні засоби для протидії сучасним і майбутнім кіберзагрозам.

Дякую за увагу!

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Олег ЛЕВЧИК

Назва: МЕТОДИ ЗАХИСТУ ВІРТУАЛЬНИХ МЕРЕЖ ВІД АТАК

Координатор: Вікторія КОРЕЦЬКА

Підрозділ: ННІТ

Коефіцієнт подібності 1:1.13

Коефіцієнт подібності 2:0

Тривога: 3

Після аналізу Звіту подібності констатую наступне:

☐ виявлені в роботі запозичення є сумлінними і не мають ознак плагіату. Тому робота визнається самостійною і допускається до захисту;

☐ виявлені в роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на доопрацювання;

☐ виявлені в роботі запозичення є недобросовісними і мають ознаки плагіату або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. У зв'язку з чим, робота не допускається до захисту.

…… ……

Дата *Підпис Наукового керівника*