Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод виявлення атак на BGP-маршрутизацію на стороні клієнта

Виконав:

студент II курсу, групи КСМм-23-1

Чухлєбов І.Я.
(прізвище, ініціали)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерні системи та мережі
(повна назва освітньої програми)

Керівник: доц. Ільїна І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Чухлєбову Ігорю Ярославовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Метод виявлення атак на BGP-маршрутизацію на стороні клієнта

затверджена наказом по університету від “ 22 ” листопада 2024 р. № 1237 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20 січня 2025 р.

3. Вхідні дані до роботи протокол BGP; вимірювання часу RTT.

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз протоколу прикордонного шлюзу;
2) аналіз сучасних підходів до захисту комп`ютерних мереж;
3) програмна реалізація;
4) проведення експериментальних досліджень;
5) висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 11 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз літератури та дослідження методів	26.11.24-30.11.24	
	атак на BGP-маршрутизацію		
2	Огляд протоколів архітектури та особливостей	02.12.24-05.12.24	
	функціонування протоколу BGP		
3	Формулювання критеріїв, та підходів для	06.12.24-10.12.24	
	детектування атак за допомогою аналізу RTT		
4	Розробка алгоритму обробки RTT для аномалій	11.12.24-21.12.24	
5	Проведення експериментів	23.12.24-03.01.25	
6	Оформлення матеріалів кваліфікаційної роботи	04.01.25-07.01.25	
7	Подання кваліфікаційної роботи керівникові	08.01.25-11.01.25	
8	Подання кваліфікаційної роботи на рецензу-	13.01.25-17.01.25	
	вання		

Дата видачі завдання 25 листопада 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Ільїна І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 63 с., 14 рис., 1 дод., 11 джерел.

BGP-МАРШРУТИЗАЦІЯ, ВИЯВЛЕННЯ АТАК, BGP HIJACKING, ROUTE LEAKS, DOS-АТАКИ, АНОМАЛІЇ МАРШРУТИЗАЦІЇ, МЕРЕЖЕВА БЕЗПЕКА, ВЕРИФІКАЦІЯ МАРШРУТІВ, ФІЛЬТРАЦІЯ МАРШРУТІВ, АВТОНОМНІ СИСТЕМИ (AS), TCP-AO, RPKI, ЗАХИСТ МЕРЕЖ.

Метою кваліфікаційної роботи є розробка методу виявлення атак на BGP-маршрутизацію зі сторони клієнта для підвищення ефективності ідентифікації аномалій у маршрутизації та забезпечення безпеки Інтернет-мереж від потенційних загроз.

У ході виконання кваліфікаційної роботи було проаналізовано основні вразливості та типи атак на BGP-маршрутизацію, такі як BGP Hijacking, Route Leaks та атаки типу відмови в обслуговуванні (DoS). Розглянуто існуючі методи виявлення атак на BGP, визначено їх переваги та обмеження. Запропоновано новий метод виявлення атак зі сторони клієнта, який орієнтований на аналіз локальної поведінки клієнтської маршрутизації та виявлення підозрілих аномалій у оновленнях маршрутів. Для перевірки ефективності запропонованого підходу було розроблено сценарії симуляцій атак на BGP, а також проведено тестування на реальних і симульованих даних. Метод включає використання алгоритмів виявлення аномалій, механізмів верифікації маршрутів та вдосконалених технік фільтрації для мінімізації хибнопозитивних спрацювань та покращення точності виявлення.

Результати роботи можуть бути застосовані у сфері мережевої безпеки для підвищення надійності Інтернет-маршрутизації, захисту від критичних збоїв та створення стійких систем комунікації.

ABSTRACT

Master’s thesis: 63 pages, 14 figures, 1 appendices, 11 sources.

BGP ROUTING, ATTACK DETECTION, BGP HIJACKING, ROUTE LEAKS, DOS ATTACKS, ROUTING ANOMALIES, NETWORK SECURITY, ROUTE VERIFICATION, ROUTE FILTERING, AUTONOMOUS SYSTEMS (AS), TCP-AO, RPKI, NETWORK PROTECTION.

The major goal of this thesis is to develop a method for detecting attacks on BGP routing from the client side to improve the efficiency of anomaly identification in routing and ensure the security of Internet networks against potential threats.

In the course of the qualification work, the main vulnerabilities and types of attacks on BGP routing, such as BGP Hijacking, Route Leaks, and Denial of Service (DoS) attacks, were analyzed. Existing methods for detecting BGP attacks were reviewed, and their advantages and limitations were identified. A novel method for detecting client-side attacks was proposed, which focuses on the analysis of local client routing behavior and the identification of suspicious anomalies in routing updates. To validate the proposed approach, simulation scenarios of BGP attacks were developed, and the effectiveness of the method was evaluated using real-world and simulated datasets. The method includes the use of anomaly detection algorithms, route validation mechanisms, and advanced filtering techniques to minimize false positives and improve detection accuracy.

The results of the work can be applied in the field of network security to enhance the reliability of Internet routing, protect against critical disruptions, and create more resilient communication systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AD – виявлення аномалій, метод виявлення відхилень у звичній поведінці BGP-трафіку для ідентифікації можливих атак (англ., Anomaly Detection)

AS – автономна система, окремий мережевий сегмент, що управляється одним адміністратором і використовує єдину політику маршрутизації (англ., Autonomous System)

AS-PATH – атрибут BGP, який показує послідовність автономних систем, через які проходить маршрут (англ., Autonomous system path)

BGP – протокол прикордонного шлюзу, що використовується для маршрутизації між автономними системами (AS) в Інтернеті (англ., Border Gateway Protocol)

Community Strings – атрибути BGP, які використовуються для передачі додаткової інформації між автономними системами (англ., Community Strings)

DDoS – розподілена атака на відмову в обслуговуванні, яка здійснюється з декількох джерел одночасно (англ., Distributed Denial of Service)

DoS – атака на відмову в обслуговуванні, спрямована на перевантаження маршрутизаторів або блокування доступу до мережевих ресурсів (англ., Denial of Service)

eBGP – зовнішній протокол BGP для обміну маршрутами між різними автономними системами (англ., External BGP)

iBGP – внутрішній протокол BGP для обміну маршрутами між маршрутизаторами в межах однієї автономної системи (англ., Internal BGP)

MED – атрибут BGP, що визначає пріоритет між декількома виходами до однієї і тієї ж автономної системи (англ., Multi-Exit Discriminator)

MoAS – ситуація, коли один і той самий IP-префікс оголошується кількома автономними системами, що може свідчити про потенційну атаку (англ., Multiple Origin AS)

MP-BGP – розширення BGP для підтримки маршрутизації декількох протоколів(англ., Multiprotocol BGP)

RPKI – інфраструктура публічних ключів для перевірки достовірності маршрутів BGP (англ., Resource Public Key Infrastructure)

RTBH – механізм, що дозволяє автоматично блокувати трафік, спрямований до певного IP-адресата, при виявленні атаки (англ., Remote Triggered Black Hole)

TCP-AO – опція аутентифікації TCP-сесій для забезпечення захисту BGP-з'єднань (англ., TCP Authentication Option)

ВСТУП

Протокол прикордонного шлюзу (Border Gateway Protocol) – це протокол, який використовується для підключення до Інтернету в наш час. Однак цей протокол є недосконалим з точки зору безпеки, що залишає користувачів беззахисними у випадку обходу їхнього трафіку. Ці недоліки протоколу впливають на конфіденційність даних, доступність зв'язку та цілісність сервісу, що зазвичай називають тріадою безпеки Confidentiality, Integrity and Availability (CIA)[1]. Незважаючи на існування деяких рішень і систем, які можуть бути використані мережевими провайдерами та деякими великими корпораціями, звичайні користувачі та компанії стикаються з недоліками, які існують по суті.

Навіть якщо користувач може виявити, що його трафік перенаправляється, він не може діяти на мережевому рівні, тому його можливості зводяться до попередження провайдера і використання більш високого рівня політик безпеки, таких як припинення вхідних і вихідних передач даних і використання шифрування в публічних службах, які ще не використовують його.

Метою цієї роботи є розробка моделі системи моніторингу, яка здатна виявляти перенаправлення трафіку протоколу прикордонного шлюзу (BGP) на основі лише часу проходження в обидва кінці (RTT) для кожної мережі, що моніториться. Моніторинг повинен базуватися на наборі датчиків, розміщених в декількох місцях, а результати вимірювань повинні передаватися в центральне місце, яке виступає в якості сервера, а також містить веб-сайт, який дозволяє підключення авторизованих користувачів.

Обробка результатів, отриманих різними датчиками, повинна бути достатньо чутливою, щоб правильно співвіднести минулі вимірювання з результатами, що надходять, і виявити стійкі послідовні відхилення від передбачуваного середнього значення.

Що стосується роботи датчиків, то вони не вимагають багато можливостей або ресурсів, що дозволяє розгортати їх на віртуальних приватних серверах (VPS), розподілених по різних дата-центрах, що є життєво важливим для забезпечення глобального покриття і правильного виявлення атак на маршрутизацію, оскільки вони можуть бути помічені тільки датчиками на достатній відстані від мережі, що моніториться, і/або точки атаки.

1 BGP-МАРШРУТИЗАЦІЮ СУЧАСНІ ТЕНДЕНЦІЇ

1.1 Протокол прикордонного шлюзу

BGP є стандартом поточної інтернет-маршрутизації і використовується як протокол де-факто[2] для з'єднання з Інтернетом. Він був побудований з використанням всього досвіду, отриманого під час розробки та використання протоколу зовнішнього шлюзу (протокол EGP не слід плутати з однойменним сімейством протоколів) і прийшов на зміну йому.

Незважаючи на те, що BGP розроблявся як протокол зовнішнього шлюзу (EGP), він також може працювати як протокол внутрішнього шлюзу (IGP), як показано на рисунку 1.1 (адаптовано з [1]), але внутрішній аспект протоколу не буде розглядатися в цій роботі, оскільки він не схильний до атак перехоплення, які будуть розглянуті пізніше. З цього моменту і надалі, коли згадується BGP, слід вважати, що мова йде про зовнішню частину протоколу.

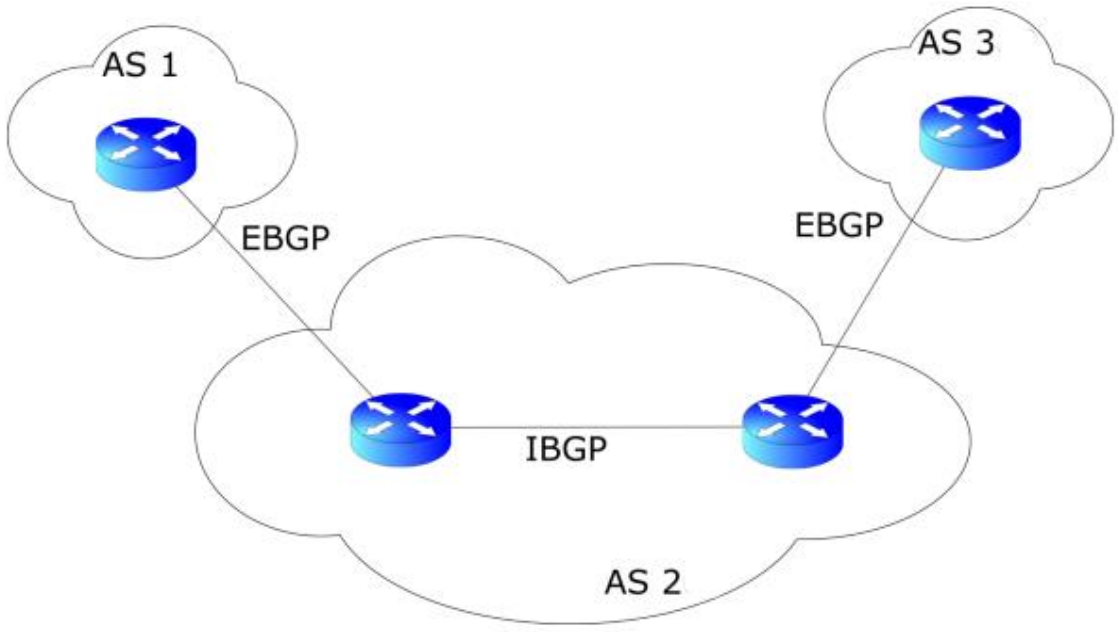

Рисунок 1.1 – Зовнішній та внутрішній BGP

У версії 4 BGP зазначено, що основною функцією BGP є механізм обміну інформацією між кількома BGP-мовними системами, також відомими як Автономні системи, які Hawkinson, J. та ін. визначили як "пов'язану групу з одного або декількох IP-префіксів, керовану одним або декількома мережевими операторами, яка має єдину і чітко визначену політику маршрутизації" в RFC 1930. Ці автономні системи (AS) ідентифікуються номером автономної системи (ASN), який може бути будь-яким числом від 1 до 64511 для публічних ASN і від 64512 до 65355 для приватних ASN. Оскільки це обмежений діапазон ASN, пізніше була опублікована пропозиція розширити ASN з 2 байт до 4 байт у форматі [крапка].

BGP працює в Інтернеті шляхом побудови таблиці досяжності і графа мережі відповідно до інформації, що міститься в повідомленнях від інших BGP-маршрутизаторів (також званих прикордонними маршрутизаторами (BR)), що містять маршрути для досягнення інших AS, залишаючи маршрутизатору-одержувачу рішення про вибір кожного маршруту на основі певних правил, таких як найбільш специфічний IP-префікс або довжина (в кількості хопів) маршруту. Ці повідомлення створюються в повідомленнях UPDATE і містять набір атрибутів, серед яких обов'язковими є такі:

- origin – звідки роутер дізнався маршрут. 0 для внутрішнього, 1 для зовнішнього і 2 для неповного або невідомого походження;
- next Hop – адреса Інтернет-протоколу (IP), яка повинна використовуватися для досягнення оголошеної AS. Якщо BGP працює як IGP, цей атрибут слід поширювати і не змінювати;
- as Path – список всіх AS, які передали оголошення і формують шлях до початкової AS. Кожного разу, коли маршрутизатор отримує оголошення і перерозподіляє його своїм одноранговим маршрутизаторам, він готує свою ASN до шляху, який вже присутній в повідомленні.

Важливим аспектом, який буде розглянуто пізніше, є те, що для того, щоб уникнути зациклення при передачі повідомлень, щоразу, коли

маршрутизатор бачить у повідомленні оголошення маршруту з його власним ASN, він просто відкидає пакет.

Будь-які дві BR, які утворюють з'єднання, в загальному випадку називаються сусідами, але з'єднання можуть бути трьох типів[4-5]:

Peer-Peer – переважно в ядрі Інтернету, ці сусіди передають трафік, не очікуючи грошової винагороди, але не повідомляють один одному маршрути транзиту або іншим одноранговим вузлам;

Customer-Provider (клієнт-провайдер) – клієнт платить провайдеру за можливість оголошувати свої маршрути і транзитний трафік решті частини Інтернету;

Sibling-Sibling – коли дві AS адмініструються однією і тією ж компанією або органом, вони можуть передавати трафік без обмежень.

Ці визначення є важливою концепцією для визначення властивості BGP-маршрутизації «valley-free», яка полягає в тому, що маршрут, який проходить через межу провайдер-клієнт або однорангову межу, не повинен перетинати іншу однорангову межу або межу клієнт-провайдер. Це важливе поняття для подальшого обговорення перехоплення BGP. Рисунок 1.2 (адаптований з [6]) ілюструє маршрут, який порушує цю властивість і є, по суті, оригінальним Proof-of-Concept, зробленим Пілосовим і Капелою.

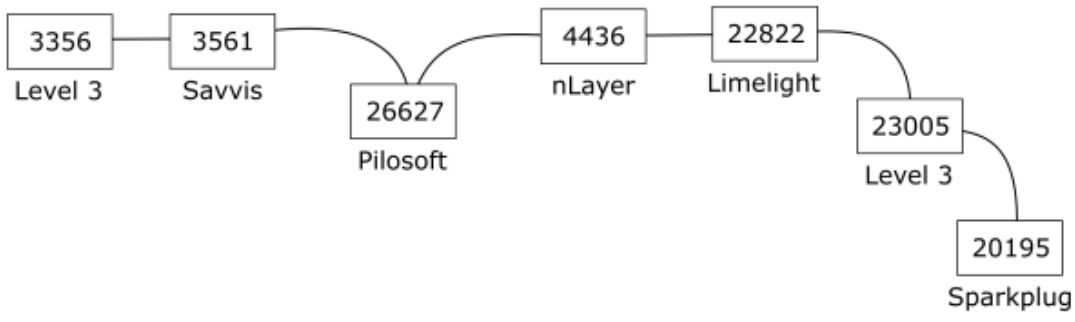

Рисунок 1.2 – Маршрут, що порушує властивість « valley-free» BGP

Як вже було сказано, релевантна інформація для створення BGP-шляхів надсилається в повідомленнях UPDATE. Ці повідомлення містять

кілька полів, які можуть використовуватися для вказівки маршрутів, які більше не прийнятні, або для додавання нових маршрутів у мережу. Нові маршрути передаються у полі, яке називається Network Layer Reachability Information (NLRI) (Інформація про досяжність мережевого рівня). Коли маршрутизатор отримує новий маршрут, він повинен додати власний ASN до AS Path, перш ніж розповсюджувати цей маршрут.

Разом із новими маршрутами маршрутизатор може також оголошувати кілька атрибутів, які не є обов’язковими загальновідомими атрибутами, описаними раніше. Один із таких важливих атрибутів – це атрибут Communities, який класифікується як Optional Transitive (необов'язковий транситивний атрибут), і він може не підтримуватися всіма реалізаціями BGP. Однак, якщо він підтримується, цей атрибут може бути використаний для BGP-хіджакінгу (захоплення маршрутів), як буде розглянуто далі.

Communities можна описати як набір маршрутів, які мають спільну властивість і згруповані для спрощення застосування політик маршрутизації [7]. Цей атрибут розроблений для роботи як 32-бітове значення, але деякі значення рекомендується вважати зарезервованими. Також існують три значення, які є глобальними для кожної автономної системи (AS):

- NO_EXPORT (0xFFFFFF01) – цей маршрут або група маршрутів не повинні передаватися стороннім учасникам BGP-конфедерації (автономної системи, розділеної на внутрішні під-AS для адміністративних цілей [8]
- NO_ADVERTISE (0xFFFFFF02) – цей маршрут або група маршрутів не повинні передаватися жодним BGP-пірінговим вузлам.
- NO_EXPORT_SUBCONFED (0xFFFFFF03) – цей маршрут або група маршрутів не повинні передаватися жодним BGP-пірінговим вузлам за межами автономної системи, навіть якщо вони є частиною BGP-конфедерації.

Далі в цьому розділі буде описано вид атаки на BGP, що використовує згадані глобальні спільноти.

1.2 BGP вразливості

З особливостей протоколу, що були представлені, стає очевидно, що BGP не був розроблений з урахуванням питань безпеки. Це створило довготривалі вразливості, які не можна повністю вирішити, а лише моніторити. Усі транзакції BGP базуються на довірі, і в останні роки спостерігається зростання випадків BGP-хіджакінгу (викрадення маршрутів), спричинених як помилковими конфігураціями, так і зловмисними намірами.

Ці проблеми становлять серйозну загрозу через низький рівень усвідомлення таких атак та значну кількість повідомлень UPDATE, що циркулюють у мережі (у 2020 році за даними опублікованими в роботі [9] повідомила про понад 65 000 повідомлень UPDATE на хвилину).

Через відсутність механізмів для автентифікації цих повідомлень будь-хто з мінімальним досвідом може сфальсифікувати оголошення BGP, вставити нові маршрути в мережу або створити хибні шляхи AS.

Однак не всі перенаправлення, що відбуваються в Інтернеті, є нелегітимними. Одною з проблем при виявленні BGP-хіджакінгу є розрізнення між абсолютно нормальними перенаправленнями, такими як інженерія трафіку або фізичні причини (збої обладнання, природні катастрофи), і ненормальними перенаправленнями (зловмисні наміри або помилкові конфігурації). Тим не менш, атаки такого роду на BGP не є першими і, безсумнівно, не будуть останніми.

Зловмисники, хакери або просто мережні фахівці вже використовують такі техніки, як розподілені атаки типу "відмова в обслуговуванні" (DDoS), коли зловмисник намагається порушити доступ користувача до системи, перевантажуючи мережне з'єднання або ресурси сервера. Також використовуються блокування систем доменних імен (DNS), такі як "Великий китайський фаєрвол", коли інтернет-провайдери (ISP) або урядові установи можуть впроваджувати хибні записи, спрямовуючи користувачів на неправильні сайти, таким чином блокуючи доступ до легітимних сайтів.

Коли йдеться про атаки на BGP, хіджакінг є загальним терміном, що охоплює три типи атак:

- Blackholing – подібно до блокування DNS, зловмисники можуть приймати трафік, призначений для жертви, і відкидати його (наприклад, на інтерфейс Null0), таким чином повністю припиняючи потік інформації;

- Impersonating – це коли зловмисники видають себе за жертву, або відповідаючи на перехоплений трафік, або використовуючи оголошені IP-префікси для надсилання спам-листів;

- Interception – головний предмет занепокоєння. Зловмисники приймають трафік, можуть його прочитати (і навіть модифікувати), перш ніж відправити до оригінального призначення. Атаки перехоплення зазвичай залишаються невидимими для жертви і є складними для виконання, оскільки під час атаки необхідно зберігати "чистий шлях" до жертви. У 2008 році Пілосов і Капела представили свою атаку Man-In-The-Middle (MITM) [10], в якій був запропонований новий спосіб створення такого "чистого шляху".

Атаки типу Interception працюють за принципом перехоплення трафіку, призначеного для жертви, подібно до Blackholing або Impersonating, але з подальшим перенаправленням трафіку до передбачуваного отримувача після його читання і/або модифікації. У 2014 році Renesys повідомила про щонайменше три відомі методи здійснення таких атак, серед яких: Man-In-The-Middle (MITM) від Пілосова та Капели, підхід, що використовує атрибут Communities у BGP, та техніка, яка експлуатує оголошення маршрутів тільки до пірів (peers), а не провайдерів, щоб скористатися блокуванням для отримання вільного транзиту. Початок атаки є переконання інших автономних систем (AS) у тому, що наше оголошення є правильним і, найголовніше, найкращим маршрутом до оголошеного IP-префіксу. Це можна досягти, вигравши умови змагання (race conditions), що відбуваються в таблицях BGP, наприклад:

- найбільш специфічний префікс (наприклад, /16 є більш специфічним, ніж /8, тому маршрутизатори обирають його);

- коротший маршрут за довжиною шляху AS (AS Path).

Існують інші атрибути, які можуть вплинути на цей вибір. Наприклад, у Cisco є Best Path Algorithm з 13 умовами, а у Juniper – з 15. Проте зазначені умови (специфічність префіксу та короткість шляху) є найбільш використовуваними під час спроби "виграти" (де "виграш" означає отримання найкращого маршруту замість легітимного оголошувача) контроль над маршрутами. [10]

Проблема виникає, коли спроба "отруїти" інші BGP-маршрутизатори здійснюється без урахування доставки трафіку до початкового пункту призначення. Некоректне оголошення може призвести до того, що всі пакети, які надсилаються, повертаються за короткий час, оскільки інші вважають, що шлях до атакованої автономної системи (AS) проходить через цей маршрут.

Пілосов і Капела використовують властивість відсутності циклів у AS Path для забезпечення "чистого" шляху до жертви. Вони аналізують результати traceroute і відзначають, який ASN потрібно додати до AS Path. Це призводить до того, що коли шлях, який повинен залишатися "чистим", отримує повідомлення **UPDATE** із викраденим маршрутом, він відкидає цей пакет, оскільки його власний ASN вже присутній у маршруті.

Останнім кроком у процесі є створення статичного маршруту до першої AS у "підробленому" AS Path, перенаправляючи трафік до початкового пункту призначення.

MITM за методом Пілосова і Капели: Легітимне оголошення

Пілосов і Капела започаткували метод Man-In-The-Middle (MITM), який використовує легітимне оголошення маршруту для забезпечення успішного перехоплення трафіку без порушення його доставки до початкового пункту призначення.

Основні етапи:

Етап 1. Легітимне оголошення маршруту жертвою:

- жертва (наприклад, as 200) оголошує свій маршрут сусіднім автономним системам;

- це оголошення виглядає абсолютно коректним і відповідає правилам bgp.

Етап 2. Розповсюдження оголошення мережею:

- сусіди жертви передають оголошення далі в мережу.

- в результаті, інші маршрутизатори отримують маршрут до ip-префіксів жертви.

Етап 3.Вибір найкращого маршруту:

- кожен маршрутизатор обирає найкращий маршрут до IP-префіксу, оголошеного жертвою, на основі правил BGP (таких як довжина AS Path або специфічність префіксу);

- це створює глобальну видимість маршруту до жертви через її сусідні автономні системи.

Чому це важливо для MITM: легітимне оголошення є першим етапом у підготовці до атаки MITM. Воно допомагає зловмиснику зберегти "чистий" шлях до жертви, використовуючи властивість відсутності циклів у AS Path. Це дозволяє атакуючому:

- перехоплювати трафік, призначений для жертви;

- перенаправляти його назад після читання або модифікації, залишаючись непомітним для кінцевого користувача.

Ця стратегія показує, наскільки важливим є врахування навіть легітимних оголошень у контексті потенційних атак.

У рисунках 1.3–1.7 описується гіпотетична атака Man-In-The-Middle:

1. AS 200, наша жертва, спочатку оголошує свій маршрут своїм сусідам (рисунок 1.3);

2. сусіди розповсюджують це оголошення по всій мережі (рисунок 1.4);

3. у результаті всі системи досягають згоди, і кожен маршрутизатор обирає найкращий маршрут (рисунок 1.5).

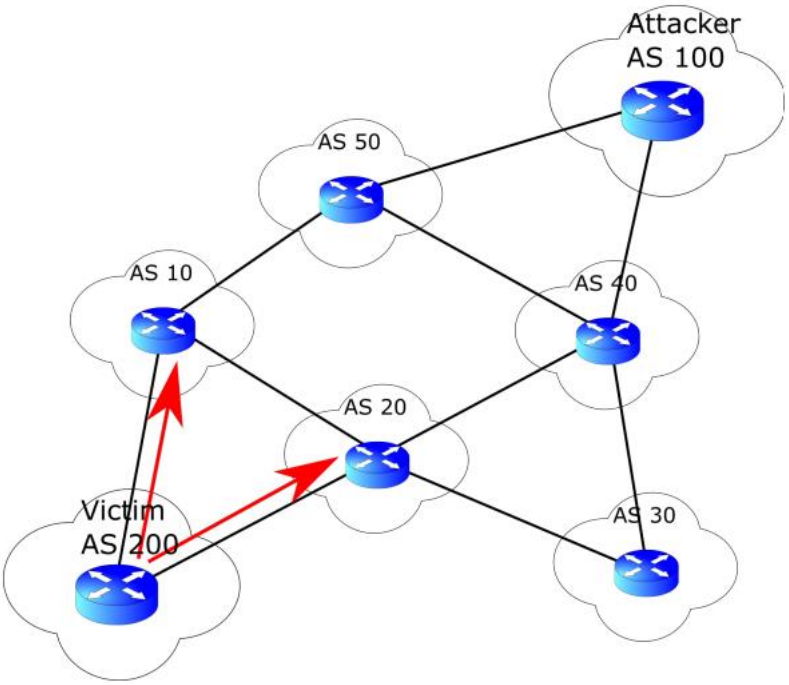

Рисунок 1.3 – MITM за методом Пілосова і Капели

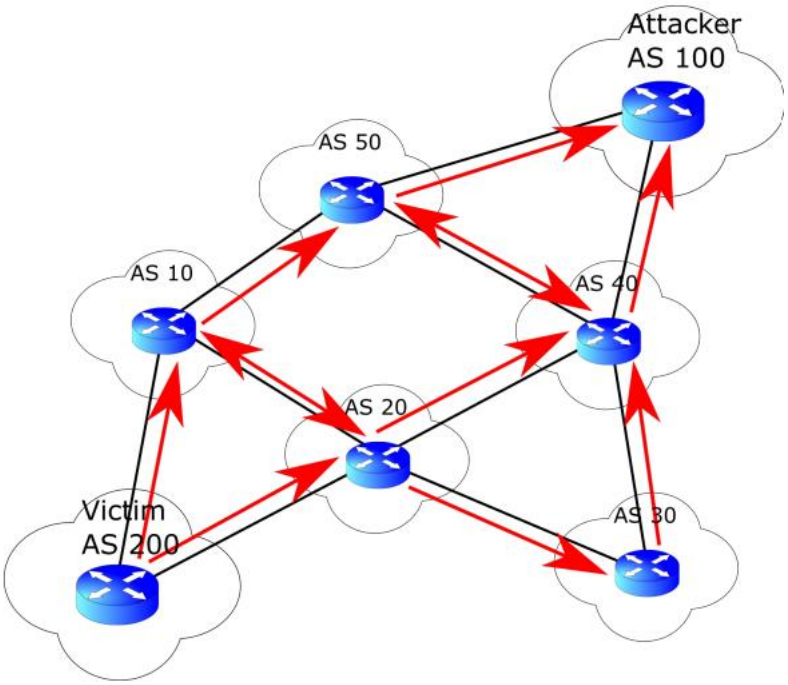

Рисунок 1.4 – MITM за методом Пілосова і Капели: Розповсюдження оголошення

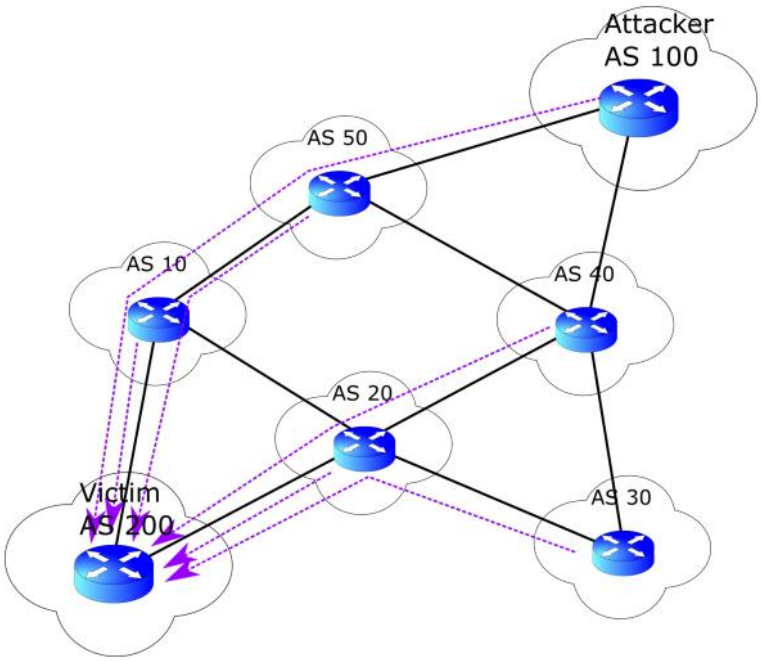

Рисунок 1.5 – MITM за методом Пілосова і Капели: Конвергенція мережі

На цьому етапі зловмисник повинен проаналізувати traceroute до цільового об’єкта, щоб підготувати зловмисне оголошення, додаючи до AS Path ASN автономних систем у маршруті. Це змусить маршрутизатори на цьому шляху відкидати пакети (рисунок 1.6).

Нарешті, після розповсюдження хибного оголошення, майже всі в мережі починають надсилати пакети, призначені для жертви, до зловмисника. Потім зловмисник перенаправляє ці пакети до жертви, використовуючи збережений "чистий" шлях (рисунок 1.7).

Попри всю підготовку перед запуском атаки, викрадення маршруту досить легко виявити за допомогою простого traceroute, який покаже адреси, через які проходив трафік, перш ніж повернутися до початкового пункту призначення. Зловмисники можуть налаштувати значення TTL (Time To Live) таким чином, щоб приховати "нові" маршрутизатори у результатах traceroute і, з певним успіхом, замаскувати свою присутність.

Цього можна досягти шляхом додавання достатньої кількості значень до TTL, щоб traceroute не показував небажані кроки між відправником і жертвою.

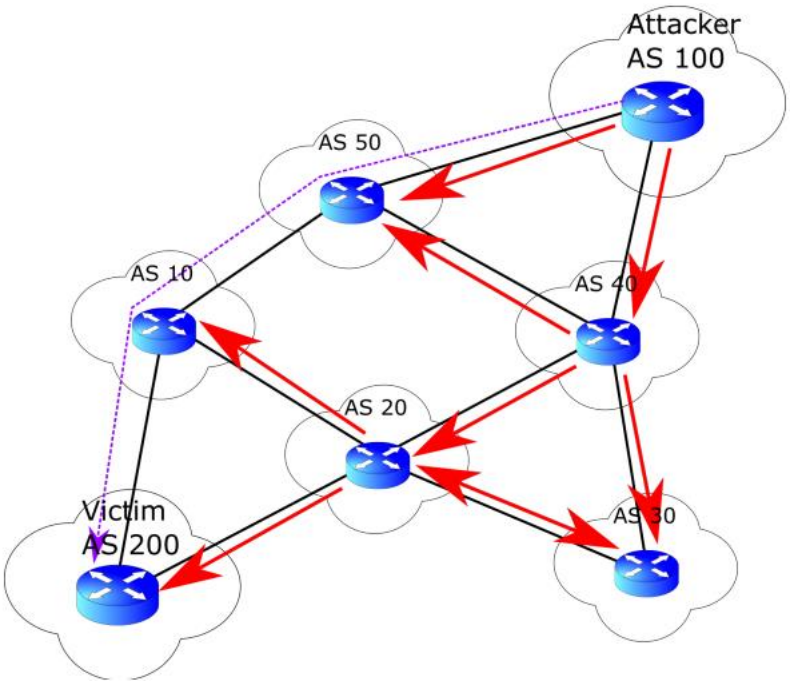

Рисунок 1.6 – MITM за методом Пілосова і Капели: Хибне оголошення

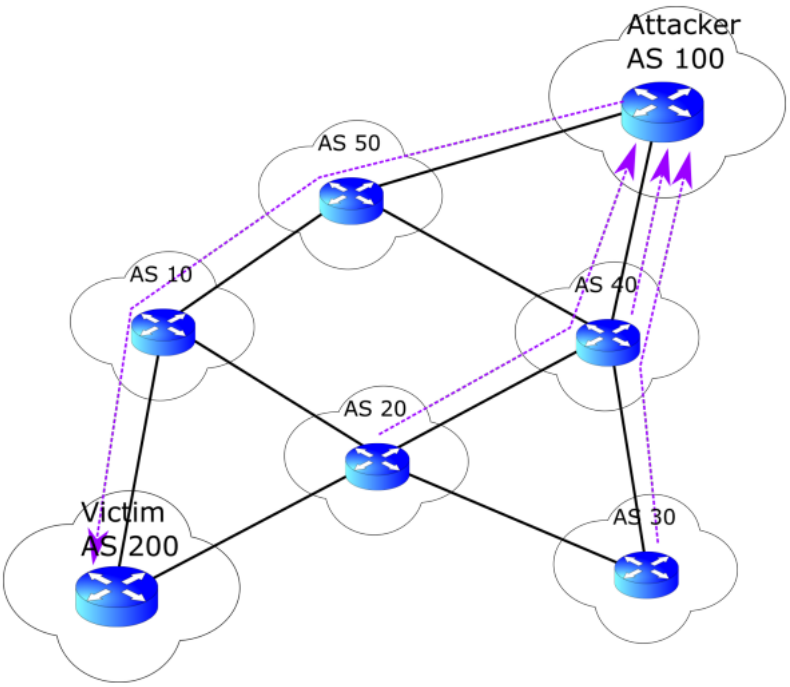

Рисунок 1.7 – MITM за методом Пілосова і Капели: Викрадення маршруту в процесі

Однак цей метод не впливає на один важливий аспект трафіку – часовий фактор. Оскільки BGP використовує сеанси Transmission Control

Protocol (TCP) для обміну даними, можливо визначити час передачі за допомогою ACK-повідомлень, що дозволяє виявити значний стрибок у часі при використанні маскування TTL, якщо перенаправлення достатньо велике, щоб бути помітним.

На рисунку 1.8 показано повний шлях, яким проходив трафік під час початкової атаки, продемонстрованої Пілосовим і Капелою, коли весь трафік, призначений для DEFCON, був перенаправлений на сервери Pilosoft у Лос-Анджелесі. При застосуванні маскування TTL, як показано на рисунку 1.9, з записів було видалено 10 кроків, але час залишився тим самим. Це робить виявлення атаки легким завданням для людини чи системи, які аналізують результати: між показниками часу видно стрибок із 28 мс до 88 мс.

У 2014 році компанія Renesys висловила думку, що цей тип виявлення все ще використовується переважно в рамках тестування або демонстрації концепції.

```
 2 12.87.94.9 [AS 7018] 4 msec 4 msec 8 msec
 3 tbr1.cgcil.ip.att.net (12.122.99.38) [AS 7018] 4 msec 8 msec 4 msec
 4 ggr2.cgcil.ip.att.net (12.123.6.29) [AS 7018] 8 msec 4 msec 8 msec
 5 192.205.35.42 [AS 7018] 4 msec 8 msec 4 msec
 6 cr2-loopback.chd.savvis.net (208.172.2.71) [AS 3561] 24 msec 16 msec 28 msec
 7 cr2-pos-0-0-5-0.NewYork.savvis.net (204.70.192.110) [AS 3561] 28 msec 28 msec 28 msec
 8 204.70.196.70 [AS 3561] 28 msec 32 msec 32 msec
 9 208.175.194.10 [AS 3561] 28 msec 32 msec 32 msec
10 colo-69-31-40-107.pilosoft.com (69.31.40.107) [AS 26627] 32 msec 28 msec 28 msec
11 tge2-3-103.ar1.nyc3.us.nlayer.net (69.31.95.97) [AS 4436] 32 msec 32 msec 32 msec
12 * * *   (missing from trace, 198.32.160.134 - exchange point)
13 tge1-2.fr4.ord.llnw.net (69.28.171.193) [AS 22822] 32 msec 32 msec 40 msec
14 ve6.fr3.ord.llnw.net (69.28.172.41) [AS 22822] 36 msec 32 msec 40 msec
15 tge1-3.fr4.sjc.llnw.net (69.28.171.66) [AS 22822] 84 msec 84 msec 84 msec
16 ve5.fr3.sjc.llnw.net (69.28.171.209) [AS 22822] 96 msec 96 msec 80 msec
17 tge1-1.fr4.lax.llnw.net (69.28.171.117) [AS 22822] 88 msec 92 msec 92 msec
18 tge2-4.fr3.las.llnw.net (69.28.172.85) [AS 22822] 96 msec 96 msec 100 msec
19 switch.ge3-1.fr3.las.llnw.net (208.111.176.2) [AS 22822] 84 msec 88 msec 88 msec
20 gig5-1.esw03.las.switchcommgroup.com (66.209.64.186) [AS 23005] 84 msec 88 msec 88 msec
21 66.209.64.85 [AS 23005] 88 msec 88 msec 88 msec
22 gig0-2.esw07.las.switchcommgroup.com (66.209.64.178) [AS 23005] 88 msec 88 msec 88 msec
23 acs-wireless.demarc.switchcommgroup.com (66.209.64.70) [AS 23005] 88 msec 84 msec 84 msec
```

Рисунок 1.8 – BGP-хіджакінг без маскування TTL

Іншою описаною технікою є використання BGP Communities для обмеження оголошень лише до провайдерів верхнього рівня (upstream), що

дозволяє обмежити викрадення маршруту певними просторовими межами. Цей підхід є більш ризикованим, оскільки потребує повного розуміння Communities і способу, яким вони агрегують і обробляють оголошення.

Суть методу полягає в оголошенні бажаних маршрутів із використанням спільноти типу P:71990, яка належить до категорії P:7DNNA . Як показують бази даних WHOIS, це тип, що запобігає поширенню префіксів. У цьому контексті:

- D вказує, які пірингові вузли блокуються в поширенні (1 – міжнародні вузли, 2 – вузли всередині країни);

- NN визначає провайдерів верхнього рівня (99 означає всіх провайдерів, інші числа представляють окремих провайдерів);

- A визначає дію і може приймати лише значення 0 (Не оголошувати префікс).

Ця техніка може бути використана для обмеження поширення маршрутів, що є важливим для локалізації впливу викрадення маршрутів.

Остання описана процедура передбачає здійснення оголошень лише до піринговых вузлів (peers), а не до клієнтів, провайдерів чи родинних вузлів (siblings). Оскільки, за визначенням, пірингові вузли не передають трафік чи перенаправляють оголошення, це забезпечує ефект, аналогічний до підходу зі спільнотами (Communities), і обмежує викрадення маршруту найближчими автономними системами.

Однак наявність знань для здійснення цих атак у мережі BGP не автоматично ставить під загрозу протокол чи його учасників. Для цього зловмисникам необхідно отримати доступ до мережі – безпосередньо або віддалено.

Безпосередній доступ можна отримати, якщо зловмисник працює поблизу фізичних маршрутизаторів або проникне у місця, де вони знаходяться.

Віддалений доступ майже завжди залежить від зловмисної атаки, яка дозволяє отримати контроль над каналом передачі даних.

Одним із найпопулярніших механізмів для цього є "Троянський кінь" – програма, яка може маскуватися під легітимний додаток для користувача, виконуючи зловмисні дії без його відома. У останні роки такі трояни стали більш специфічними, враховуючи нові способи використання Інтернету, що сприяло появі "соціально спроектованих троянів".

Окрім троянів, зловмисники також можуть використовувати:

- помилки чи збої у програмному забезпеченні на комп’ютерах;
- фішингові атаки, здійснені через електронну пошту;
- інші форми зловмисного коду, такі як віруси, черв’яки та шкідливе ПЗ (malware).

Зазначені вище механізми та атаки розглядалися з теоретичної точки зору. Проте, за даними Dyn Research (раніше Renesys), 2013 рік став роком, коли ці атаки вперше були зафіксовані в Інтернеті, з більш ніж 150 містами, де щонайменше одна жертва зазнала таких атак. З того часу викрадення маршрутів, як навмисні, так і випадкові, фіксуються майже щомісяця у різних місцях, включно з Україною.

Одним із найбільш серйозних випадків сталося в березні 2013 року, коли Vega (український телекомунікаційний провайдер) почав викрадати префікси, що належали Агентству з атомної зброї Великої Британії (AWE). Трафік, який мав напряму з'єднувати Х'юстон (Техас, США) і Великобританію, почав проходити через Франкфурт (Німеччина) і Київ (Україна), перш ніж повернутися до початкового пункту призначення. Викрадення тривало близько 90 хвилин і включало понад 150 префіксів, зокрема префікси AWE. Крім цього, Vega протягом 5 днів оголошувала хибні префікси, що включали компанії, такі як Walmart, Pepsi Co., Royal Mail, The Football Association, банки, телекомунікаційні компанії та інші.

Інші випадки були також зафіксовані тією ж компанією (Dyn Research), серед яких міжнародний трафік.

Трафік з Гвадалахари, Мексика, до Вашингтона, округ Колумбія, США, який перенаправлявся через Москву (Росія), і Мінськ (Білорусь), перед поверненням до Америки.

Трафік з Чикаго, Іллінойс, США, до Тегерану, Іран, проходив складний маршрут, перш ніж досягти початкового пункту призначення (рисунок 1.9).

Рисунок 1.9 – Приклад викраденого маршруту

Водночас, були зафіксовані випадки викрадення маршрутів у межах одного міста. У 2013 році трафік, який мав проходити між двома точками у Денвері, Колорадо, США, був значно відхилений і обрав маршрут через Лондон, Велика Британія, та Рейк’явік, Ісландія. Це був один із епізодів масштабного викрадення, зафіксованого в період з 31 липня по 19 серпня 2013 року, яке походило з Ісландії й траплялося 17 разів.

1.3 Захист BGP

Що можна зробити, щоб запобігти викраденню маршрутів? Вже було запропоновано та частково впроваджено кілька рішень, які варіюються від оновлених протоколів (BGPSEC, SBGP, So-BGP) до криптографічних

підписів у пакетах (RPKI) або простих методів, які можуть бути реалізовані провайдерами у власній інфраструктурі, наприклад, фільтрація префіксів.

Усі згадані механізми можна вважати активними рішеннями, але також існують методи пасивного моніторингу, спрямовані на відстеження Інтернету та BGP-оголошень з метою виявлення помилкових повідомлень UPDATE.

Найрадикальнішим рішенням було б змінити протокол на новий або оновлений, який міг би підтримувати перевірені оголошення або криптографічно підписані пакети. Уже були зроблені спроби створення таких рішень.

Secure BGP (S-BGP) – розробка розпочалася у 1997 році і має на меті трирівневий метод, який використовує інфраструктуру відкритих ключів (PKI), новий транзитивний атрибут BGP і IPsec для автентифікації та шифрування пакетів. PKI складається з різних механізмів, політик і процедур, що підтримують два основні компоненти: сертифікаційні центри (Certification Authorities) і криптографію з відкритим ключем, щоб забезпечити безпечну комунікацію для людей чи організацій через Інтернет. Ця інфраструктура, застосована до S-BGP, автентифікує IP-блоки, які оголошуються, а також власника AS і маршрути BGP у повідомленнях UPDATE. Новий атрибут використовується для включення цифрових підписів для підтвердження дійсності оголошення разом із інформацією PKI. Нарешті, IPsec забезпечує належне шифрування пакетів, що проходять через Інтернет, забезпечує цілісність даних і автентифікацію між маршрутизаторами BGP.

Secure Origin BGP (soBGP) – ще одна пропозиція, розроблена інженерами Cisco, отримала назву Secure Origin BGP. Вона базується на концепції цифрових сертифікатів, використовуючи три типи сертифікатів для забезпечення безпечної комунікації між вузлами BGP, і додає новий тип повідомлення до BGP, який називається SECURITY message. Це повідомлення передає всю інформацію, що стосується сертифікатів, і створює зворотну сумісність, оскільки не змінює жодного існуючого типу

повідомлень. За допомогою цих повідомлень вузли soBGP обмінюються сертифікатами: Entity Certificates (щоб підтвердити ідентичність вузла), Authorization Certificates (щоб призначити IP-блоки маршрутизаторам) і Policy Certificates (щоб передати політики для префіксів між маршрутизаторами мережі) [35]. Хоча S-BGP є більш повним і надійним рішенням, soBGP є значно легшим, існує компроміс між безпекою та часом обробки, який мають врахувати учасники, зацікавлені в цих протоколах.

BGPsec – найновіший метод BGPsec базується на деяких ідеях S-BGP, зокрема використанні криптографічних ключів для забезпечення правильної передачі маршрутів через вузли BGP. Використовуючи інфраструктуру відкритих ключів (RPKI), BGPsec пропонує спосіб захисту поширення маршрутів, коли кожен вузол підписує свої повідомлення, створюючи набір вкладених підписаних повідомлень при поширенні оголошень через Інтернет. RPKI також забезпечує зв’язок між ASN та IP-блоками, формуючи ланцюг довіри разом із підписаними повідомленнями. Після включення своїх блоків до RPKI автономна система (AS) може асоціювати Route Origination Authorization (ROA), тобто спеціальний підписаний блок, виданий RPKI для конкретної AS, із оголошенням. ROA, підписані вкладені повідомлення та RPKI створюють простий, але надійний метод захисту AS Path і забезпечення деякого рівня безпеки протоколу.

Ці рішення були розроблені для застосування на стороні мережі, контрольованій інтернет-провайдерами (ISP), але, поки вони знаходяться у процесі розробки, існують інші заходи, які можна впровадити. Одним із таких прикладів є фільтрація префіксів або маршрутів, коли адміністратори автономних систем (AS) можуть впроваджувати правила, які забороняють їхнім маршрутизаторам приймати оголошення, що містять помилкову або підозрілу інформацію. Базове правило полягає у фільтрації префіксів, які належать самій AS, але навіть це іноді ігнорується.

Крім цього, автономні системи можуть почати фільтрування з використанням баз даних, таких як Internet Routing Registry (IRR), які

покривають деякі, але не всі випадки, забезпечуючи певний рівень асоціації між ASN і IP-блоками. Однак ці бази даних є загальнодоступними, і будь-хто може внести в них дані, що робить їх не повністю надійними.

Навіть якщо адміністратори не фільтрують вхідний трафік, вони можуть почати фільтрувати вихідний, щоб запобігти поширенню помилкових конфігурацій з їхнього боку. Якщо всі учасники почнуть фільтрувати свій трафік, кількість випадків ненавмисного викрадення маршрутів значно зменшиться. Проте, як зазначають Пілосов і Капела: "Доки кожен не фільтруватиме ідеально кожного, ці двері залишатимуться відкритими" .

Щодо дій, які можуть бути виконані на стороні інтернет-провайдерів (ISP), існують деякі заходи, які, однак, зазвичай вимагають співпраці двох або більше пірів або надмірної кількості ручної роботи. Наприклад, використання TCP MD5 Authentication дозволяє двом маршрутизаторам обмінюватися повідомленнями BGP (оскільки BGP працює через TCP) із асоційованим корисним навантаженням, яке хешується за допомогою алгоритму MD5. Приймаючий маршрутизатор може хешувати вміст повідомлення таким самим способом і порівняти результат із хешем, що вже міститься в повідомленні. Якщо результати відрізняються, пакет має бути відхилений, оскільки йому не можна довіряти. Однак це вимагає ручного налаштування наборів ключів для кожної пари пірів і періодичної зміни цих ключів, щоб мінімізувати ризик атак. Такий підхід створює значний обсяг роботи для мережевих адміністраторів, що є бар'єром для широкого впровадження .

Інше рішення, яке вже розглядалося в контексті BGPsec, – це використання IPsec для забезпечення безпечної комунікації між сусідніми маршрутизаторами. Це може бути реалізовано за допомогою Authentication Header (AH) або Encapsulating Security Payload (ESP), які забезпечують автентифікацію та/або шифрування даних. Однак цей метод вирішує лише локальні проблеми безпеки і не запобігає викраденню маршрутів.

Нарешті, механізм Generalised TTL Security Mechanism (GTSM) може

бути використаний для запобігання атакам із використанням маніпуляції значенням TTL (розглядалося раніше у контексті атаки MITM Пілосова і Капели). Оскільки BGP-сесії зазвичай використовують TTL = 1 (оскільки їхні сусіди знаходяться на відстані одного переходу), це значення легко змінити для приховування слідів атаки. GTSM встановлює значення TTL = 255 (максимальне для 8-бітного числа), і коли маршрутизатор отримує пакет із TTL менше 254, він відхиляє його. Однак, як і в попередніх випадках, це рішення не запобігає викраденню маршрутів у масштабах усієї мережі, а є лише невеликим інструментом у вирішенні більш глобальної проблеми.

Окрім рішень, які передбачають використання мережевого та/або транспортного рівнів моделі взаємодії відкритих систем (OSI), протягом років були представлені підходи, які зазвичай є різновидами систем моніторингу для BGP-оголошень. Для таких рішень важливо розрізняти підхід на основі інформації про дані (data-plane) та підхід на основі управління (control-plane), залежно від того, яку інформацію вони використовують для визначення необхідності попередження власника AS.

Data-plane характеризується контролем мережі та ухваленням рішень на основі попередньо визначених політик, навіть якщо дані не передаються. Натомість control-plane орієнтований на керування трафіком і пакетами, коли вони фактично надходять на маршрутизатор. У рамках згаданих рішень можуть існувати механізми, які моніторять Інтернет на основі чинних політик, даних, що передаються через BGP, або навіть обох підходів.

Використання лише інформації control-plane забезпечує легкість впровадження рішення, але може бути неточним під час аналізу мережі через відсутність даних BGP. Використання лише data-plane також дозволяє створити відносно просту систему, але її ефективність обмежується кількістю та розташуванням моніторингових датчиків. Нарешті, метод, який поєднує обидва підходи, може виконувати спільне виявлення викрадень у реальному часі, але його точність також обмежується кількістю та розташуванням датчиків.

2 СУЧАСНІ ПДХОДИ ДО ЗАХИСТУ КОМП`ЮТЕРНИХ МЕРЕЖ

2.1 Системи виявлення вторгнень у комп'ютерні мережі

Системи виявлення вторгнень (СВВ) є одним із ключових інструментів забезпечення інформаційної безпеки в сучасних комп'ютерних мережах. Їх основна мета полягає в моніторингу мережевого трафіку та виявленні ознак потенційно зловмисних дій, таких як спроби проникнення, експлуатації вразливостей або викрадення даних. Завдяки цим системам організації можуть своєчасно реагувати на загрози та мінімізувати ризики, пов’язані з інформаційною безпекою.

СВВ виконують низку важливих функцій, серед яких:

- виявлення несанкціонованого доступу до мережевих ресурсів;
- аналіз та ідентифікація аномальної активності у трафіку;
- виявлення атак, спрямованих на експлуатацію вразливостей;
- формування звітів та попереджень про інциденти безпеки;
- надання інформації для прийняття рішень щодо усунення або запобігання загроз.

Системи виявлення вторгнень класифікуються за різними ознаками, зокрема за принципом виявлення загроз, архітектурою та типом аналізу.

За методом виявлення:

- сигнатурні системи (Signature-Based Detection). Ці системи використовують базу даних шаблонів атак (сигнатур), яка постійно оновлюється. Перевагою є висока точність виявлення відомих атак, однак вони неефективні проти нових або модифікованих загроз;
- системи виявлення аномалій (Anomaly-Based Detection). Порівнюють поточну активність у мережі з базовою нормою. Аномальні відхилення сигналізують про можливі загрози. Такі системи здатні виявляти нові типи атак, але можуть мати високий рівень хибних спрацьовувань;

- гібридні системи (Hybrid Detection). Поєднують сигнатурний підхід та аналіз аномалій, забезпечуючи більш збалансований підхід до виявлення загроз.

За місцем розташування:

- мережеві системи виявлення вторгнень (Network-Based Intrusion Detection Systems, NIDS). Моніторять мережевий трафік у реальному часі, аналізуючи пакети, що проходять через певні точки в мережі. Їх перевага полягає в тому, що вони можуть забезпечити захист усієї мережі;

- хостові системи виявлення вторгнень (Host-Based Intrusion Detection Systems, HIDS). Працюють безпосередньо на кінцевих пристроях, моніторячи журнали подій, файли та активність процесів. Вони забезпечують детальну інформацію про стан окремого пристрою.

За способом реагування:

- пасивні системи. Генерують сповіщення про потенційні загрози, але не вживають жодних заходів для їх усунення;

- активні системи (системи запобігання вторгнень, IPS). Мають можливість автоматично блокувати зловмисну активність, зокрема шляхом відключення мережевого з'єднання або блокування доступу.

Попри значну ефективність СВВ, їх впровадження може супроводжуватися низкою проблем:

- висока кількість хибних спрацьовувань, що ускладнює реагування;

- необхідність постійного оновлення баз сигнатур та профілів аномалій;

- значне навантаження на мережеві ресурси при аналізі великих обсягів трафіку;

- складність інтеграції з іншими системами захисту.

Сучасні дослідження та розробки у сфері СВВ спрямовані на використання машинного навчання та штучного інтелекту для підвищення точності виявлення загроз. Також активно впроваджуються хмарні рішення для моніторингу розподілених мережевих середовищ. Інтеграція СВВ із

системами управління інформаційною безпекою (SIEM) дозволяє забезпечити більш комплексний підхід до захисту даних.

Таким чином, системи виявлення вторгнень є невід'ємною складовою кіберзахисту сучасних мереж, забезпечуючи своєчасне виявлення загроз і зниження ризиків інформаційної безпеки.

2.2 Системи протидії вторгненням

Системи протидії вторгненням (СПВ) є критично важливим елементом сучасних засобів забезпечення інформаційної безпеки, які розширюють функціонал систем виявлення вторгнень. Головною відмінністю СПВ є їх здатність не лише виявляти загрози, але й активно запобігати їх реалізації. Це досягається завдяки можливості автоматичного блокування підозрілих дій або реагування на загрози у реальному часі.

СПВ забезпечують багаторівневий підхід до захисту мережі, поєднуючи аналіз трафіку, моніторинг активності на кінцевих пристроях та впровадження політик безпеки. Їх основна мета – забезпечити безперервність роботи мережі, мінімізуючи ризики втрати даних, компрометації ресурсів або порушення нормальної роботи систем.

СПВ виконують широкий спектр завдань:

- моніторинг трафіку в реальному часі. Аналіз мережевого трафіку на предмет підозрілих дій, таких як сканування портів, спроби несанкціонованого доступу або аномальна активність;
- запобігання атакам. Блокування шкідливого трафіку до того, як він досягне цільової системи, зокрема під час атак типу DDoS, SQL-ін’єкцій або експлуатації вразливостей;
- ізоляція заражених пристроївю Відключення або обмеження доступу пристроїв, які можуть бути скомпрометовані, для уникнення поширення загрози в мережі;

- застосування політик безпекию Автоматизація виконання правил доступу, обмеження мережевих з’єднань відповідно до встановлених норм.

СПВ працюють шляхом безперервного аналізу мережевого трафіку, активності користувачів і роботи програмного забезпечення на пристроях. Використовуючи заздалегідь налаштовані політики, вони приймають рішення про блокування або дозвіл певної активності. Зазвичай такі системи інтегруються з іншими засобами захисту, такими як міжмережеві екрани, антивірусне ПЗ та системи управління подіями безпеки (SIEM), забезпечуючи багаторівневий підхід до безпеки.

СПВ використовують кілька основних методів для виявлення загроз:

- сигнатурний аналіз. Порівняння трафіку або активності з базою даних відомих шаблонів атак (сигнатур). Цей метод ефективний для виявлення відомих загроз, але вразливий до атак нульового дня;

- аналіз аномалій. Виявлення відхилень від встановлених норм поведінки мережі або системи. Цей підхід дозволяє виявляти нові типи загроз, але може спричиняти хибні спрацьовування;

- контекстуальний аналіз. Врахування додаткової інформації, такої як час, місце чи тип активності, для ухвалення рішень про безпеку;

- машинне навчання. Використання алгоритмів для аналізу великих обсягів даних і визначення прихованих закономірностей, що можуть свідчити про загрози.

Попри значні переваги, впровадження СПВ пов'язане з низкою викликів:

- хибні спрацьовування. Через надто чутливі налаштування системи можуть блокувати легітимний трафік, що впливає на нормальну роботу мережі;

- високе навантаження на ресурси. Аналіз великого обсягу трафіку в реальному часі може викликати значне навантаження на апаратне забезпечення;

- складність налаштування. Розробка ефективних політик безпеки та налаштування системи потребують значного рівня експертизи;

- інтеграція. Необхідність забезпечити сумісність СПВ із іншими засобами безпеки в мережі.

Сучасні СПВ все більше інтегрують технології штучного інтелекту (ШІ) та машинного навчання для підвищення точності виявлення загроз і зменшення кількості хибних спрацьовувань. Інтеграція з хмарними технологіями дозволяє ефективно моніторити розподілені мережі та забезпечувати масштабованість. Крім того, поява концепцій, таких як Zero Trust Security, стимулює розвиток СПВ у напрямку більш гнучких і адаптивних систем, здатних реагувати на нові виклики в кіберпросторі.

Системи протидії вторгненням є невід’ємною частиною сучасного підходу до забезпечення кібербезпеки. Їх використання дозволяє суттєво зменшити ризики для критичних інформаційних активів і забезпечити стійкість мереж до загроз різного рівня складності.

3 АРХІТЕКТУРА СИСТЕМИ

3.1 Запропоноване рішення та архітектура

У цій кваліфікаційній роботі пропонується рішення, яке базується на сервері, побудованому за модульною архітектурою, і змінній кількості датчиків (Probes). На сервері різні модулі обробляють дані та взаємодіють один з одним для забезпечення моніторингу BGP. Це працює у поєднанні з датчиками, розташованими по всьому світу, які запускають скрипти, завантажені через вебсайт, і виконують необхідні тести.

Більш детально підхід до рішення представлено на рисунку 3.1, де показано компоненти, які вже впроваджені, їх зв’язки та протоколи чи мови, використані для цього.

Модуль користувачів (User Module) зберігає всю інформацію про користувачів, перевіряє їхні облікові дані та дозволи для надання доступу до частин системи, а також передає ці дозволи іншим компонентам.

Вебсайт і графічний інтерфейс користувача (GUI) використовують функціональність зазначеного модуля для забезпечення можливості для авторизованих користувачів із необхідними дозволами переглядати, редагувати, додавати або видаляти дані із системи.

Компонент зберігання (Storage), пов’язаний із вебсайтом, але також використовуваний іншими компонентами, дозволяє зберігати скрипти, завантажені на сервер. Компонент логування (Logging) зберігає всі журнали, згенеровані різними компонентами в процесі їхньої роботи.

Центральним елементом всієї системи, який пов’язаний майже з усіма іншими компонентами, є база даних (Database). Вона зберігає всю інформацію про датчики, скрипти, результати, тривоги та будь-які дані, згенеровані внаслідок їх взаємодії.

Компонент управління датчиками (Probe Management) відповідає за всі аспекти, пов’язані з датчиками, зокрема створення нових екземплярів у системі, завантаження скриптів на датчики, перевірку їхньої працездатності та виконання інших завдань. Цей компонент взаємодіє з датчиками за допомогою Secure Shell (SSH) і Secure Copy (SCP) (через SSH-канал), забезпечуючи безпечний спосіб комунікації між ними.

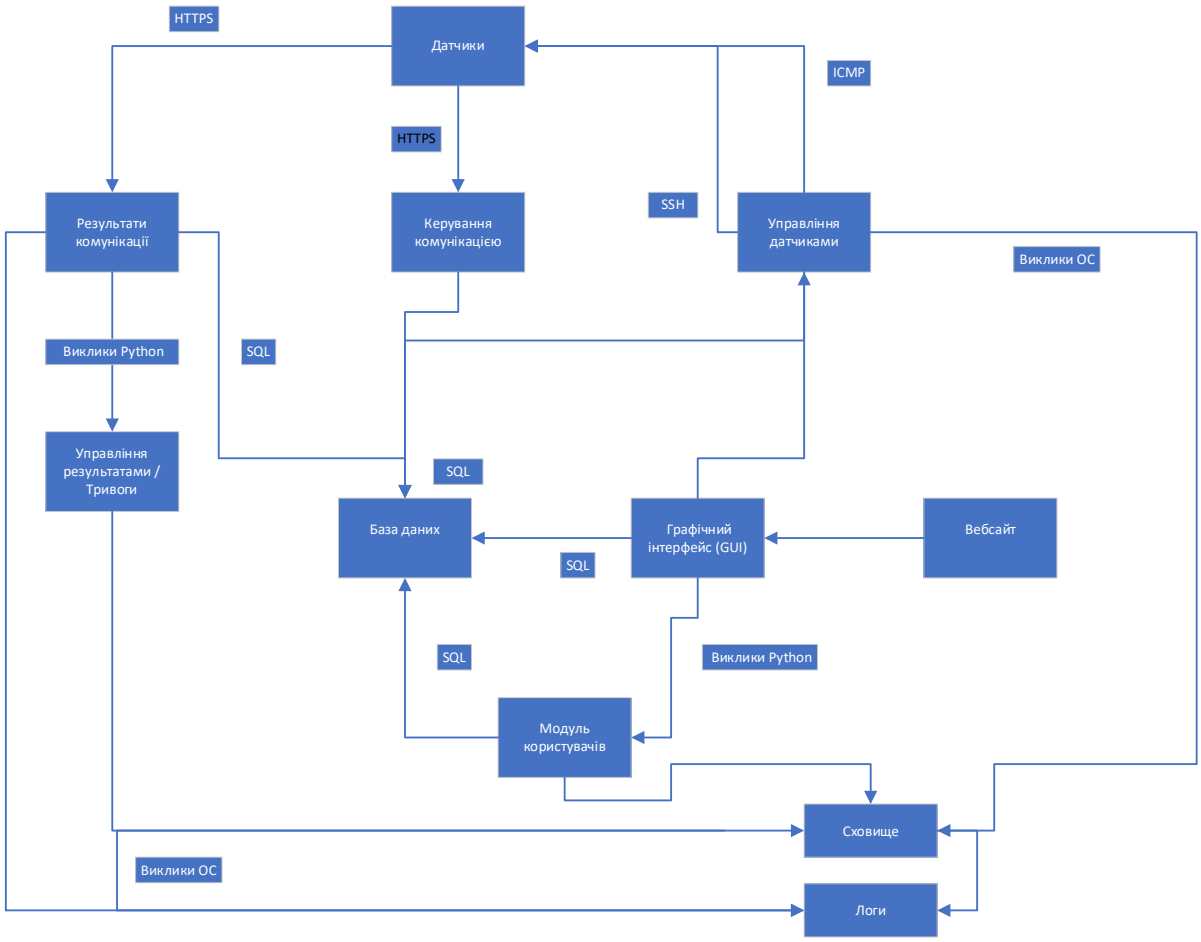

Рисунок 3.1 – Схема компонентів системи

Після налаштування датчиків і запуску скриптів на них необхідно забезпечити на сервері кінцеві точки для правильної прийому даних. Для цього два компоненти відповідають за зв’язок із датчиками. Перший компонент – керування комунікацією – відповідає за прийом даних, пов’язаних із використанням датчиків та їх ресурсів. Цей компонент отримує

сигнали від скриптів, які здійснюють моніторинг датчиків, і може вносити зміни до даних у базі даних за необхідності. Другий компонент, який також виступає як кінцева точка для зв'язку з датчиками, – це результати комунікації. Він відповідає за збір даних, отриманих датчиками під час виконання тестів, і збереження їх у базі даних. Після збереження даних результати комунікації має сповіщати компонент керування комунікацією про те, що новий результат готовий для аналізу. Компонент результати комунікації, який також включає підкомпонент тривоги, відповідає за обробку вхідних результатів через компонент результати комунікації. На основі отриманих даних він встановлює відповідні тривоги або для конкретної цілі, або для датчика, який зібрав результат. Усі ці компоненти знаходяться на серверній стороні системи, проте для ефективної роботи їм потрібен набір датчиків, які надають значущу інформацію. Кількість датчиків може бути змінною, але важливо, щоб вони були географічно віддаленими один від одного для забезпечення глобального покриття. Це особливо важливо в контексті атак на маршрутизацію BGP, оскільки датчики, розташовані занадто близько до місця атаки, можуть бути «сліпими» до її виникнення.

3.2 Модель даних

Для зберігання всіх даних, необхідних для проєкту, була спроєктована модель даних, яка враховує потреби кожного компонента. Протягом проєкту таблиці та стовпці в кожній таблиці змінювалися для відображення поточного стану розробки. Діаграма, представлена на рисунку 3.2, є реляційною діаграмою, що показує таблиці та зв'язки між ними. У цій діаграмі вже можна побачити специфічні для реалізації концепції, такі як таблиці Django, тому представлені типи даних були згенеровані Django для кращої адаптації до визначення моделі в MariaDB.

Варто зазначити, що всі таблиці мають числовий ідентифікатор (ID), який слугує первинним ключем для цієї таблиці. Починаючи з цього моменту, це поле є неявним при обговоренні таблиці.

Починаючи з таблиць, які створюються за замовчуванням при старті проєкту Django і використовувалися в цьому проєкті, варто згадати три таблиці. Таблиця Group зберігає назву кожної групи користувачів. Група – це простий спосіб надання дозволів певному набору користувачів, замість того щоб робити це індивідуально для кожного. Зв'язана з групою таблиця User Groups асоціює ідентифікатор групи (Group ID) з ідентифікатором користувача (User ID) і зберігає інформацію про те, які користувачі входять до яких груп. Нарешті, таблиця User зберігає всі дані, що стосуються користувачів системи, наприклад, пароль, дату останнього входу, ім’я, прізвище, ім’я користувача та електронну пошту, які вказуються під час реєстрації, а також дату приєднання користувача до системи. Вона також містить три поля для перевірки, чи вважається користувач суперкористувачем, співробітником або чи є він активним.

Таблиця User є фундаментальною частиною цієї моделі, оскільки встановлює права володіння в різних таблицях моделі. Однією з таких таблиць є таблиця Probe, яка зберігає всю інформацію, що стосується датчиків, підключених до системи, зокрема текстовий ідентифікатор, IPv4 та IPv6 адреси, статус датчика (Активний або Резервний), користувача, якому належить датчик, і поле, що вказує, чи зустрічався датчик з аномалією. Таблиця Backup, пов’язана з таблицею Probe, зберігає два ідентифікатори датчиків і створює зв’язок типу «Активний-Резервний» між двома датчиками.

Поряд із датчиками дві додаткові таблиці зберігають інформацію про різні типи скриптів, які можна завантажити через вебсайт: одна для скриптів обробки, а інша для моніторингових скриптів. Перша зберігає всі скрипти, які використовуватимуться на сервері для обробки вхідних результатів і генерації тривог, тоді як друга зберігає скрипти, що надсилаються до

датчиків, або для їх ініціалізації, або для запуску на датчику з метою вимірювання різних аспектів цього проєкту. Ці таблиці мають деякі спільні стовпці: скрипт характеризується заголовком і шляхом до файлу, який містить скрипт, полем, яке визначає, чи вважається скрипт активним або неактивним, і набором параметрів, які використовуються під час виконання скриптів. За замовчуванням під час завантаження скрипта користувач має можливість вибрати ім’я сервера, IP-адресу датчика та список цілей, які будуть додані до команди виконання, але також є поле для зберігання додаткових параметрів, які можуть бути важливими для конкретного скрипта. Скрипт, подібно до датчика, має власника, тому є поле для зберігання його ідентифікатора. Нарешті, таблиця для зберігання моніторингових скриптів містить додаткове поле для розрізнення скриптів, які використовуються для ініціалізації датчика, і тих, які повинні виконуватися та проводити тести. У зв’язку з цими скриптами дві додаткові таблиці, активні скрипти обробки та активні моніторингові скрипти, зберігають пари ідентифікаторів датчиків і скриптів, щоб надати системі інформацію про те, де саме запускати ці скрипти.

Коли моніторингові скрипти виконуються на датчиках, вони збирають певні вимірювання, які зберігаються в таблиці Results. Ця таблиця наразі зосереджена на вимірюваннях RTT (Round Trip Time) і зберігає позначку часу, коли було отримано результат, а також мінімальне, максимальне, середнє значення та відхилення, отримані під час вимірювання. Вона також містить інформацію про те, який датчик зібрав результат, яка була ціль вимірювання, який тип результату було зібрано і якому користувачеві належить цей результат. Є також поле для позначення, чи вважається результат аномальним компонентом управління результатами. Для зберігання різних типів результатів, які можуть існувати, додаткова таблиця зберігає їхні назви і пов’язана з полем type у таблиці Result.

У попередньому абзаці згадувалася ціль (Target). Ця таблиця зберігає інформацію про користувача, який додав ціль, і IPv4-адресу цієї цілі.

Нарешті, дві таблиці зберігають інформацію щодо тривог. Таблиця Alarm Target зберігає дані про конкретну ціль, включаючи час тривоги, який датчик виконав вимірювання, і яка була ціль. Таблиця Alarm Probe містить інформацію про те, які датчики збирають аномальні вимірювання, і зберігає лише позначку часу та ідентифікатор датчика.

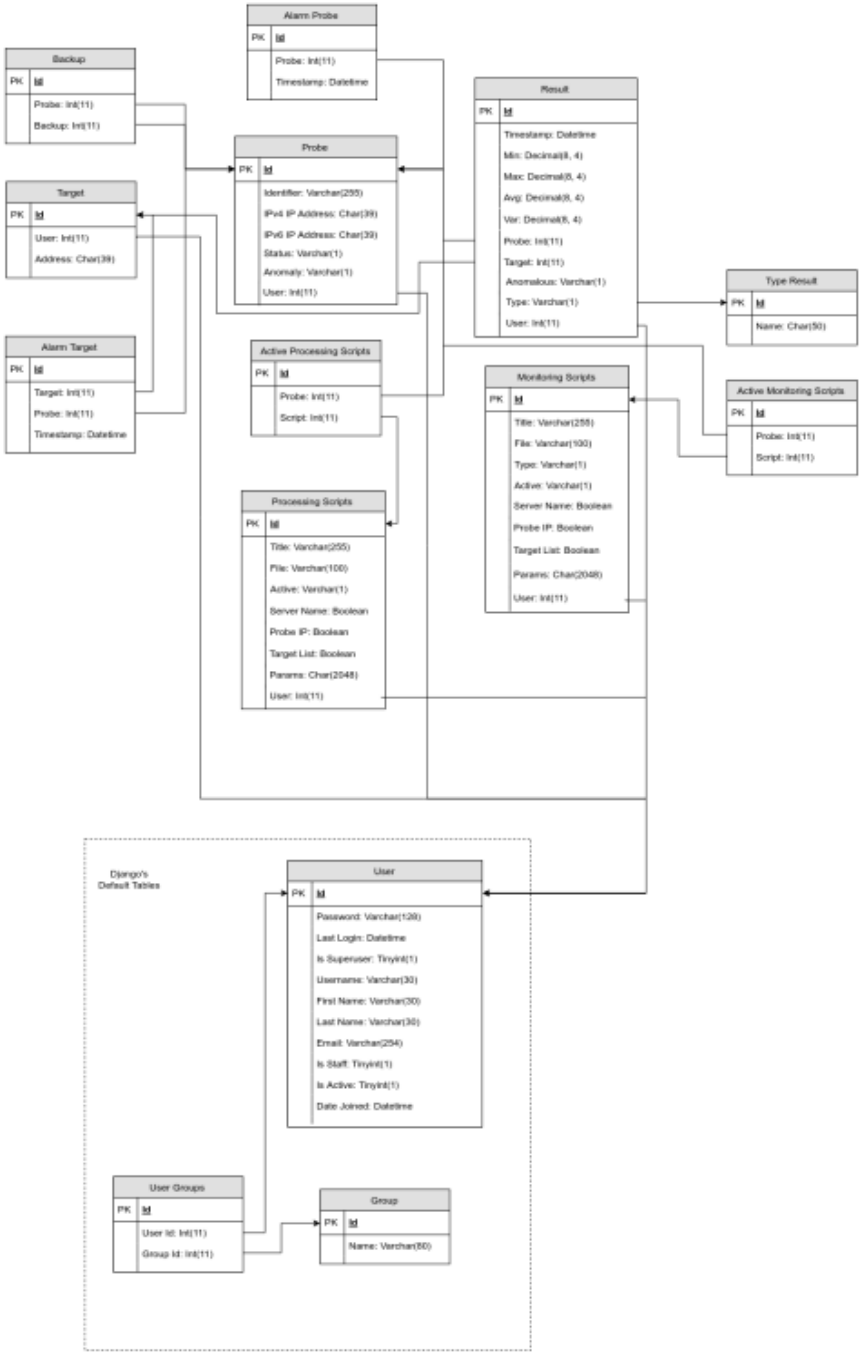

Рисунок 3.2 – Модель даних

3.3 Реалізація

Для реалізації компонентів та моделі даних, розглянутих у попередніх розділах, як основну мову програмування було використано Python версії 2.7, разом із фреймворком Django.

Django можна описати як "високорівневий веб-фреймворк для Python, що сприяє швидкій розробці та чистому, прагматичному дизайну", орієнтований на розробників, які хочуть швидко налаштувати свій вебсайт із базовими функціональностями. Під час першого розгортання проєкту Django вже містить такі функції, як реєстрація та автентифікація користувачів, панель адміністратора та підключення до бази даних, що дозволяє розробнику зосередитися на інших функціях, які відрізняють його систему.

У Django існують концепції проєкту та додатків. Проєкт – це початкова папка, створена Django, яка містить файл налаштувань (Settings), що зберігає всі необхідні конфігурації для системи, такі як конектор і облікові дані бази даних, встановлені додатки, доступні проміжні програмні забезпечення (middleware) тощо, а також Python-файл для мапінгу уніфікованих ідентифікаторів ресурсів (URLs) до відповідних додатків. Додаток, у свою чергу, є вебдодатком, який розробник намагається реалізувати. Один проєкт може містити кілька додатків, і один додаток може бути розгорнутий у кількох проєктах.

Для створення цієї програми важливо розуміти концепції Models, Views та Templates і те, як вони можуть бути зв’язані між собою за допомогою файлів URL, які є в проєкті та додатку. Це фактично схоже на шаблон проєктування Model-View-Controller (MVC), де "модель – це дані, подання – це вікно на екрані, а контролер – це зв’язок між ними". Проте команда Django визначає себе як фреймворк Model-Template-View (MTV), де View відповідає за те, які дані видно, а не за те, як вони виглядають.

Model – це набір інструкцій, які визначають, як зберігаються дані, їхні назви та типи, а також інші функції, які можна визначити за потреби. Після

визначення моделі об’єктно-реляційний відображувач (ORM) Django переводить їх у мову запитів для бази даних, що використовується. За замовчуванням Django підтримує лише реляційні бази даних, але існують сторонні розширення, які можуть забезпечити підключення до деяких нереляційних баз даних, таких як MongoDB або Google App Engine. Простими словами, моделі дозволяють розробникам писати свої моделі даних за допомогою Python-коду.

View-функції, або просто Views, – це Python-функції, які забезпечують логічну сторону додатка. Їхнім вхідним параметром є запит (Request), а вихідним – відповідь (Response), яка може варіюватися від помилки 404 до зображення чи, найчастіше, HTML-вмісту вебсторінки. Саме View-функції збирають необхідні дані через запити (querysets), які надаються моделями, і передають їх у шаблон (Template). Після створення View і встановлення зв’язків із моделями та шаблонами розробник повинен задекларувати URL, який викликатиме цю View, щоб зробити її доступною.

Нарешті, Templates створюють те, що користувачі бачать на вебсайті, заповнене інформацією, надісланою через View у поєднанні з кодом, що є в них. Ці HTML-скрипти можуть використовувати, але це не є обов’язковим, шаблонний рушій (templating engine), щоб динамічно рендерити інформацію, передану через View. Django містить власний шаблонний рушій Django Template Language (DTL), але також підтримує Jinja2 за замовчуванням. Як і у випадку з базами даних, інші шаблонні рушії можуть бути підтримані за допомогою сторонніх розширень. Файли шаблонів підтримують усі HTML-теги та/або теги шаблонного рушія, що дозволяє використовувати JavaScript, CSS та інші корисні інструменти. Це створює файл, який може містити статичну інформацію в поєднанні з циклами, умовними операторами або об’єктами з бази даних для динамічного складання відповіді, яка буде показана на вебсторінці.

База даних підтримується системою MariaDB – це система з відкритим вихідним кодом, створена розробниками оригінальної MySQL після її

придбання компанією Oracle. MariaDB має на меті бути сумісною з бінарними файлами MySQL, що робить її легкою для заміни навіть у випадку, якщо початкова розробка була здійснена на базі MySQL. MariaDB підтримується Django за замовчуванням, і завдяки використанню ORM Django та вбудованих функцій завдання написання SQL-запитів повністю знімається з розробників. Таким чином, усі операції, що взаємодіють із таблицями, стовпцями чи рядками бази даних, реалізуються на Python.

Під час створення нового проєкту Django, залежно від бібліотек або проміжного програмного забезпечення (middleware), які можуть бути завантажені у файл налаштувань (Settings) проєкту, без втручання розробника створюється приблизно 10 таблиць. Ці таблиці зберігають інформацію про користувачів, групи, дозволи, сесії та інші дані. У цьому проєкті вони використовувалися як модуль користувачів (User Module), що дозволило уникнути необхідності створювати його з нуля.

На основі цих двох компонентів у системі було визначено три групи користувачів із такими назвами та дозволами:

- звичайний користувач (Regular User): Ця група користувачів може лише увійти в систему, змінювати деталі свого облікового запису та переглядати список існуючих датчиків (Probes), скриптів (Scripts) і тривог (Alarms);

- адміністратор сервера (Server Admin): Додатково до дозволів звичайного користувача, адміністратор сервера може додавати, редагувати або видаляти датчики та скрипти;

- суперкористувач (Superuser): Ця група створюється в самому Django, а не в системі. Суперкористувачі мають доступ до адміністративної панелі Django (Admin Dashboard) і можуть керувати користувачами та групами в модулі аутентифікації та авторизації (Authentication and Authorization, який є за замовчуванням у Django). Вони також можуть керувати датчиками, скриптами, результатами та тривогами в серверному модулі.

Суперкористувач має бути створений за допомогою термінала Linux у папці, де знаходиться файл manage.py, за допомогою наступної команди: python manage.py createsuperuser.

Після введення інформації щодо імені користувача, електронної адреси та пароля для цього суперкористувача створюється новий обліковий запис із достатніми привілеями для доступу до вебсторінки за адресою https://test.com/admin. Варто зазначити, що оскільки суперкористувачі створюються через командний рядок, а не через вебсайт, після створення вони повинні бути інтегровані в групу Server Admin, щоб мати ті самі дозволи на перегляд даних.

Як було зазначено раніше, система містить два компоненти, які забезпечують кінцеві точки для взаємодії датчиків (Probes) із сервером. I Result Communication, і Management Communication побудовані на основі Django REST Framework (DRF) — користувацького додатка Django, який описує себе як "потужний і гнучкий інструментарій для створення веб-API". DRF використовувався для підтримки операцій, що відповідають архітектурі REST (Representational State Transfer).

DRF за замовчуванням пропонує вбудований браузерний API, може бути пов’язаний із базою даних, яка використовується в проєкті Django, для реалізації автентифікації користувачів для REST-запитів, серіалізації даних на основі JSON і кількох рівнів складності в їхніх класах. Це дозволяє розробникам використовувати прості GET або POST запити для побудови власної логіки в API.

Management Communication отримує інформацію від скриптів, які відстежують використання ресурсів на датчиках, і може вносити зміни до статусу датчика в базі даних. Ця кінцева точка знаходиться за адресою https://test.com/rest/Probes/, але повинна бути викликана як PUT-запит до https://test.com/rest/Probes/ із JSON у тілі запиту наступного формату:

Лістинг 3.1 – Запит до менеджера комунікацій

```
{
" ipv4_ip_address ": < string >,
" status ":
}
```

Ідентифікатор датчика (Probe ID) в URL у поєднанні з IP-адресою у тілі запиту гарантує, що в базі даних буде змінено саме правильний датчик. Поле Status вказує, чи датчик активується (1), чи деактивується (0). Якщо для датчика, який деактивується, існує резервний датчик (Backup Probe), він буде встановлений як активний, щоб забезпечити моніторинг у цій області.

Інший компонент, керування результатами, відповідає за прийняття нових результатів, запис їх у базу даних і сповіщення компонентів керування результатами і тривога про те, що новий результат потребує обробки. Датчики можуть отримати доступ до цього компонента через URL: https:// test.com/rest/results/ за допомогою POST-запиту, який містить JSON із наступною структурою:

Лістинг 3.2 – Звернення через URL: https:// test.com/rest/results/ за допомогою POST-запиту

```
1 {
2 " op ": <int >,
3 " Probe ": < string >,
4 " target ": < string >,
5 " timestamp ": < datetime >,
6 " min ": < decimal >,
7 " max ": < decimal >,
8 " avg ": < decimal >,
9 " var ": < decimal >
10 }
```

Де поле "Op" вказує на операцію, виконану датчиком. На цьому етапі проєкту підтримуються лише вимірювання RTT, що означає, що поле "op" завжди повинно мати значення 1. Інші значення відхиляються сервером. У

майбутньому можуть бути додані інші опції для підтримки інших типів тестів. Поля "probe" та "target" повинні містити IP-адреси у вигляді рядків: перше поле ідентифікує датчик, який здійснює вимірювання, а друге визначає мережу, яка моніторилася під час цього вимірювання.

Інші п’ять полів є очевидними: поле "timestamp" містить дату і час проведення вимірювань, а "max", "min", "avg" та "var" представляють максимальне, мінімальне, середнє значення і дисперсію пінгів, які були виконані датчиком до цілі (Target).

Після збереження результату в базі даних компонент Result Communication сповіщає Result Management через виклики Python, передаючи ідентифікатор результату (Result ID) як частину запиту.

Важливою частиною системи є сприйняття Сервером стану різних датчиків (Probes). Це досягається за допомогою компонента Probe Management, який відповідає за створення нових записів датчиків у базі даних, їх редагування або видалення, а також завантаження скриптів для ініціалізації датчика або виконання тестів. **Probe Management** також виконує роль з’єднання між графічним інтерфейсом користувача (GUI), який потребує взаємодії з користувачем, і рештою системи, що працює в автоматизованому режимі.

Після того як користувач завантажує файл, що містить Python- або Bash-скрипт, на вебсайт, це запускає компонент **Probe Management**, який починає виконання завдання, пов’язаного з цим скриптом. Завдання, або фонові функції, – це фрагменти Python-коду, які виконуються "за лаштунками", поки інші функції також працюють. Наприклад, у цьому проєкті рендеринг вебсайту продовжується навіть під час виконання завдання. Ці завдання реалізовані за допомогою Celery — розподіленої черги завдань, яка використовує передачу повідомлень для виконання операцій у реальному часі, але також підтримує планування завдань.

Черга завдань розроблена для розподілу роботи (завдань) між кількома потоками або машинами, при цьому виконавці (workers) завжди слухають

чергу до надходження нового завдання. Для передачі завдань Celery використовує брокера, який вирішує питання комунікації між системою Celery і виконавцями завдань. Брокер підтримує широкий спектр опцій. У цьому проєкті використовується Redis, сховище структур даних, яке можна налаштувати для роботи як черга повідомлень, кеш або навіть база даних. Redis класифікується як база даних ключ-значення (нереляційна модель).

З урахуванням цього, компонент Probe Management має три основні завдання: два з них відповідають за передачу скриптів між сервером і датчиками, а третє виконує перевірку відповідей від датчиків. Останнє завдання – це простий процес, який надсилає запити ICMP (протокол керування повідомленнями в Інтернеті), відомі як пінги, кожні 30 хвилин і перевіряє, чи відповідає датчик. Перші два завдання поділяються на:

- функцію, що надсилає один скрипт на кілька датчиків, яка використовується після завантаження нового скрипта для вимірювань;

- функцію, що надсилає кілька скриптів на один датчик, яка використовується після створення нового датчика в системі та потреби його ініціалізації.

Ці з’єднання встановлюються за допомогою бібліотеки paramiko для Python, яка надає допоміжні функції для відкриття SSH-каналу до датчика. Після цього відкривається канал SCP через транспортний рівень, наданий SSH, і команда формується з параметрами, збереженими в базі даних, перед виконанням через той самий SSH-канал.

Оскільки реальні датчики, використані в рамках цієї роботи, працюють на різних дистрибутивах операційних систем на базі Linux із різними пакетними менеджерами, інсталяційні скрипти, створені в рамках проєкту, повинні враховувати всі можливі варіанти.

Останній із компонентів, що працює автономно, відповідає за обробку вхідних результатів (Results) та створення тривог (Alarms) відповідно до скриптів, наявних на сервері. Цей компонент отримує сповіщення від Result

Communication кожного разу, коли новий результат зберігається в базі даних, і запускає нове завдання для обробки цього результату.

У межах цього завдання здійснюється запит до бази даних для перевірки, чи завантажував користувач скрипт для обробки результатів. Якщо скрипт відсутній, використовується стандартний алгоритм розрахунків. Якщо новий користувач або розробник бажає реалізувати власний метод обробки результатів, база даних може бути запитана шляхом імпорту відповідних моделей та отримання необхідних даних із бази.

ORM Django надає можливість отримати один результат, усі результати або будь-яку кількість між ними:

Лістинг 3.3 – Запит до ORM Django

```
from .models import Result
results = Result.objects.all()  # Отримує всі результати
results = Result.objects.filter(user=user_id)  # Отримує
результати користувача з цим ID
result = Result.objects.get(id=new_result_id)  # Отримує
результат із цим ID
```

Різні способи отримання даних із бази даних можна знайти в документації Django. З іншого боку, після виконання обчислень скрипт повинен мати можливість зберігати інформацію назад у базу даних, щоб встановлювати тривоги. Як і раніше, необхідно виконати імпорт для завантаження відповідних таблиць, а потім змінити або створити нові записи в цих таблицях:

Лістинг 3.4 – Отримання даних із бази даних

```
from .models import Result, Alarm_Target, Alarm_Probe

# Отримує результат із заданим ID
result = Result.objects.get(id=result_id)

# Позначає оброблений результат як аномальний
```

```
result.anomalous = '1'

# Зберігає зміни в базі даних
result.save()

from django.utils import timezone

# Створює нову тривогу для зазначеної цілі та датчика
new_alarm = Alarm_Target(target=target_id, Probe=Probe_id,
timestamp=timezone.now())

# Зберігає нову тривогу в базі даних
new_alarm.save()

# Створює нову тривогу для зазначеного датчика
new_alarm = Alarm_Probe(Probe=Probe_id,
timestamp=timezone.now())

# Зберігає нову тривогу в базі даних
new_alarm.save()
```

Метод save слід викликати після внесення всіх змін, щоб забезпечити їх збереження в базі даних.

3.3 Вебсайт та графічний інтерфейс користувача (GUI)

Для забезпечення взаємодії користувачів із системою було розроблено вебсайт із графічним інтерфейсом користувача (GUI) за допомогою інструментів, наданих фреймворком Django. Цей вебсайт слугує точкою входу для введення даних і одночасно дозволяє користувачам переглядати інформацію про тести, які виконуються. Якщо користувач має відповідні дозволи, він може отримати доступ до адміністративної сторінки для управління користувачами та групами в системі.

Кожна сторінка вебсайту вимагає автентифікації користувача, перенаправляючи його на форму входу, за винятком головної сторінки (Index), яка показана на рисунку 3.3. Вона вітає користувача і пропонує йому увійти в систему.

Форма входу є простою і містить два поля: одне для імені користувача та інше для пароля. Після підтвердження введених облікових даних Django і

модуль користувачів (User Module) перевіряють їх і надають або відхиляють доступ до вебсайту.

Рисунок 3.3 – Головна сторінка вебсайту

Після завершення процесу входу користувач отримує доступ до різних опцій, які пропонує вебсайт, через панель, розташовану у верхній частині сторінки. Ця панель, детально показана на рисунку 3.4, поділяється на розділи: Probes, Scripts і Alarms. Перші два розділи розкриваються, показуючи повний набір доступних посилань.

Рисунок 3.4 – Верхня панель вебсайту

У розділі Probes користувач із повними привілеями побачить такі опції: "Add a Probe" (Додати датчик), "See the List of Probes" (Переглянути список датчиків) і "Update Targets" (Оновити цілі).

У розділі Scripts доступні опції: "Upload a Script" (Завантажити скрипт), "See where Scripts are running" (Переглянути, де виконуються скрипти) і "See a List of Scripts" (Переглянути список скриптів).

Нарешті, Alarms – це єдине посилання без додаткових опцій. На рисунках 3.5 продемонстровані ці опції.

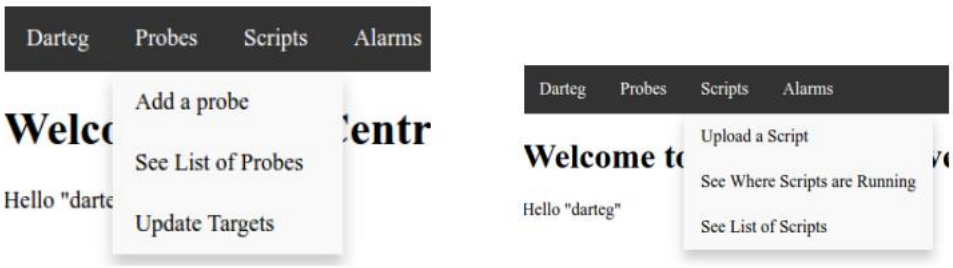

Рисунок 3.4 – Опції, доступні в панелі вебсайту

Пояснюючи кожен пункт детальніше, якщо користувач вибирає опцію "Add a Probe" (Додати датчик), відображається форма з полями для створення ідентифікаційного тега (Identifier) для датчика та введення IP-адреси, де розташований датчик. Додатково потрібно вибрати одну з двох опцій, щоб класифікувати доданий датчик як резервний (Backup) або активний (Active). Після підтвердження даних користувачем компонент Probe Management запускається для забезпечення правильної ініціалізації датчика. Якщо користувач обирає "List of Probes" (Список датчиків), відображається список, що містить ID у базі даних, ідентифікаційний тег, IP-адресу датчика та його статус. Якщо користувач має права Server Admin, також відображаються посилання для редагування (Edit Probe) та видалення датчика. Опція "Edit Probe" використовує ту саму форму, що й для додавання датчика, але поля вже заповнені. Якщо редагування було виконане помилково, користувач може повернутися до списку датчиків без збереження змін. Опція "Delete Probe" (Видалити датчик) відображає попередження, де користувач повинен підтвердити видалення. Сам процес видалення виконується через Django.

Остання опція "Update Targets" (Оновити цілі) завантажує список цілей поточного користувача з бази даних і відображає його у текстовому полі, де кожна ціль розташована в окремому рядку. Користувач може редагувати цей список або видалити його повністю.

Переходячи до опцій для скриптів, якщо користувач вибирає "Upload a Script" (Завантажити скрипт), з’являється нова порожня форма з сімома

полями для заповнення. Одне текстове поле для введення назви, поле для вибору файлу скрипта з файлової системи користувача, дві опції для класифікації скрипта як активного (Active) або неактивного (Inactive) (неактивний скрипт не буде виконуватися), і три опції щодо типу виконання скрипта, поділені на "Probe init" (Ініціалізація датчика), "Probe run" (Запуск датчика) і "Result analysis" (Аналіз результатів). Ця класифікація допомагає серверу визначити роль кожного завантаженого скрипта.

Якщо скрипт також є активним, скрипт типу "Probe init" надсилається на новододаний датчик за допомогою опції "Add a Probe", скрипт типу "Probe run" надсилається на всі активні датчики після завантаження, а скрипт "Result analysis" зберігається на сервері для виконання, коли від датчика надходить новий результат. Усі завантажені скрипти записуються в систему скриптів на сервері в папку "upload", розташовану в кореневому каталозі проєкту. Якщо доданий скрипт є типу "Probe run", завдання для його виконання встановлюється негайно після завантаження.

Останні три поля – це прапорці, які вказують, чи має скрипт будь-які параметри, що додаються до команди виконання. Наразі ці опції впливають лише на Python-скрипти та включають ім’я сервера, IP-адресу датчика та список цілей користувача.

Схоже на функціонал для датчиків, також існує опція "List of Scripts" (Список скриптів), яка відображає список із назвами та типами скриптів. У цьому списку для кожного скрипта є опції для перегляду його вмісту, редагування, виконання (якщо скрипт є типу "Probe run") або видалення, якщо користувач належить до групи Server Admin. Скрипти відображаються у групах за їхнім типом у порядку "Probe init", "Probe run" і "Result analysis".

Функція "Show Script" (Показати скрипт) викликає функцію, створену на рівні моделі, яка аналізує вміст скрипта построчно та відображає його разом із назвою та шляхом до файлу. Аналогічно до датчиків, опція "Edit Script" використовує ту ж форму, що й для створення, але без можливості вибору нового файлу скрипта. Це означає, що якщо користувач помилково

завантажив неправильний скрипт, йому потрібно завантажити новий. Також опція "Delete Script" показує повідомлення з підтвердженням, що видалення скрипта є бажаною дією, повторюючи поведінку функції "Delete Probe".

Якщо користувач уже завантажив скрипт, але з якоїсь причини його потрібно знову запустити на активних датчиках, вибір опції "See where Scripts are running" перенаправить його на нову сторінку, де можна вибрати потрібний скрипт. Після вибору система опитує кожен датчик, щоб перевірити, чи скрипт дійсно виконується, і повертає результат у двох таблицях: "Running" (Виконується) і "Not Running" (Не виконується).

Після цього користувач має можливість переміщати датчики з однієї таблиці в іншу та змінювати параметри запуску для конкретного скрипта. Після внесення всіх змін кнопка внизу сторінки дозволяє користувачу підтвердити зміни. Більшість функцій і зовнішній вигляд цієї сторінки були створені за участі Маріо Піни.

Нарешті, розділ "Alarms" збирає інформацію про виявлені тривоги, відображаючи, який датчик викликав тривогу, для якої цільової мережі вона була встановлена, і час її виникнення. Якщо тривог немає, користувачу показується повідомлення про цю ситуацію.

У випадку, якщо користувач є суперкористувачем, йому доступна адміністративна сторінка Django. На цій сторінці, яка створюється за замовчуванням під час ініціалізації проєкту Django, користувач може отримати доступ до всіх записів, що зберігаються в базі даних, редагувати їх і створювати нові. Це дозволяє суперкористувачам керувати зареєстрованими в системі користувачами та їх дозволами, залежно від груп, до яких вони належать, а також контролювати дані, які зберігаються в базі даних різними учасниками системи.

ВИСНОВКИ

У ході цієї кваліфікаційної роботи була запропонована модель системи, що здатна моніторити глобальний протокол за допомогою недорогого і простого в розгортанні рішення. Через недоліки протоколу BGP виникла необхідність створення системи, яка надавала б інформацію користувачам, що не мають доступу до BGP-рівня своєї мережі. Це стосується всіх домашніх користувачів і значної частини корпоративних клієнтів.

Якщо використати 20 різних VPS, розташованих на трьох континентах, сервер і набір датчиків, які постійно відстежують час передачі пакетів через Інтернет і намагаються виявляти аномальні стрибки у часі передачі (RTT). Тиким чином можна легко відстежувати атаки на протокол BGP.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Edwards, Nicholas, Sara Bliss Kiser, and Janel Bell Haynes. "Answering the Cybersecurity Issues: Confidentiality, Integrity, and Availability." Journal of Strategic Innovation and Sustainability 15.4 (2020).

2. Raynor, Justin, et al. "The state of the art in BGP visualization tools: A mapping of visualization techniques to cyberattack types." IEEE Transactions on Visualization and Computer Graphics 29.1 (2022): 1059-1069.

3. Teare, Diane. Designing for Cisco Internetwork Solutions (DESGN)(Authorized CCDA Self-Study Guide)(Exam 640-863). Pearson Education, 2007.

4. Nguyen, Van-Linh, Po-Ching Lin, and Ren-Hung Hwang. "Web attacks: defeating monetisation attempts." Network Security 2019.5 (2019): 11-19.

5. Moriano, Pablo, Raquel Hill, and L. Jean Camp. "Using bursty announcements for detecting BGP routing anomalies." Computer Networks 188 (2021): 107835.

6. Hansen, James D., and Thomas A. Buckhoff. "To catch a thief." Journal of Accountancy 189.3 (2000): 43.

7. Krenc, Thomas, Robert Beverly, and Georgios Smaragdakis. "AS-level BGP community usage classification." Proceedings of the 21st ACM Internet Measurement Conference. 2021.

8. Noor, Noor Maizura Mohamad, Nadiah Yayao, and Sumazly Sulaiman. "Effectiveness of using Cisco Packet Tracer as a learning tool: A case study of routing protocol." Computer software 514 (2018): 689-9.

9. Ylli, Enkli, and Julian Fejzaj. "Man in the Middle: Attack and Protection." RTA-CSIT. 2021.

10. Conti, Mauro, Nicola Dragoni, and Viktor Lesyk. "A survey of man in the middle attacks." IEEE communications surveys & tutorials 18.3 (2016): 2027-2051.

11. Чухлєбов І.Я., Ільїна І.В. ,МЕТОД ВИЯВЛЕННЯ АТАК НА BGP-МАРШРУТИЗАЦІЮ ЗІ СТОРОНИ КЛІЄНТА // Проблеми інформатизації : XII міжнародна науково-технічна конференція. – 21-22 листопада 2024. – с.104. doi: https://doi.org/10.32620/PI.24.t2