НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ВИЗНАЧЕННЯ ПАРАМЕТРІВ ЗАХИСТНИХ СИСТЕМ БЛОКУВАННЯ ЗА ДОПОМОГОЮ АЛГОРИТМУ БІНАРНОГО ПОШУКУ»

Виконав: здобувач 4 курсу
Рівень бакалавр
Галузь знань 12 «Інформаційні технології»
спеціальність 125 «Кібербезпека»
Єрохін Ілля Олександрович

Керівник: проф., д.т.н., професор
Казакова Надія Феліксівна

Рецензент: к.т.н., доцент
Ахмаметьєва Ганна Валеріївна

Одеса – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

Факультет **кібербезпеки та інформаційних технологій**
Кафедра **кібербезпеки**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **125 Кібербезпека**
Назва освітньої програми: **Кібербезпека**

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки
професор С.А. Горбаченко
_____ «____ »_____ 20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Єрохіну Іллі Олександровичу

1. Тема роботи: «Визначення параметрів захистних систем блокування за допомогою алгоритму бінарного пошуку»
Керівник роботи: к.т.н, доцент Віктор Дмитрович Бойко
затверджені наказом НУ ОЮА від «__» _____ 20___ року № _____
2. Строк подання здобувачем роботи «__» _____ 20___ рік
3. Дата видачі завдання«__» _____ 20___ рік

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка

Здобувач _____

(підпис) (прізвище та ініціали)

Керівник роботи _____

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Визначення параметрів захистних систем блокування за допомогою алгоритму бінарного пошуку» на здобуття першого(бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, освітньою програмою: Кібербезпека, містить 23 рисунки, 1 таблицю, 3 додатоки, 8 літературних джерел за переліком посилань.

Робота виконана на (37) сторінках загального тексту і (24) сторінках основного тексту.

У процесі роботи було проаналізовано стратегії та алгоритми атаки та захисту серверів, зокрема алгоритм бінарного пошуку, вивчено особливості та принципи роботи fail2ban. Для реалізації було вибрано программи та бібліотеки, проаналізовано та протестовано модель серверу, та описано його характеристики. Обрано алгоритм бінарного пошуку для дослідження ефективності його використання під час атаки на сервер захищений fail2ban.

Реалізація проводилась з використанням мови програмування Python, середовища PyCharm та бібліотеки paramiko, программ OpenBSD ssh server, fail2ban 0.11 2-2

Результати досліджують показують високу ефективність алгоритму бінарного пошуку у порівнянні з іншими способами та використовують значно менше ресурсів.

Результати роботи можуть бути використані при проектуванні або налаштуванні засобів захисту як частини системи захисту серверу, а також можуть слугувати для академічних досліджень у сфері безпеки веб-додатків або вивчення методів атак на веб-додатки.

ABSTRACT

Qualification Work on the Topic "Determination of Parameters for Blocking Protection Systems Using the Binary Search Algorithm" for Obtaining the First (Bachelor's) Level of Higher Education in Specialty 125 Cybersecurity and Data Protection, Educational Program: Cybersecurity

The work contains 23 figures, 1 table, 3 appendices, and 8 literary sources according to the reference list.

The work is completed on 37 pages of general text and 24 pages of main text.

In the course of the work, strategies and algorithms for server attacks and defenses were analyzed, specifically the binary search algorithm, as well as the features and principles of fail2ban. Programs and libraries were selected for implementation, the server model was analyzed and tested, and its characteristics were described. The binary search algorithm was chosen to study its efficiency during an attack on a server protected by fail2ban.

The implementation was carried out using the Python programming language, the PyCharm environment, the paramiko library, OpenBSD ssh server programs, and fail2ban 0.11 2-2.

The research results demonstrate the high efficiency of the binary search algorithm compared to other methods, utilizing significantly fewer resources.

The results of the work can be applied in the design or configuration of protection tools as part of a server protection system, and they can also be used for academic research in the field of web application security or the study of attack methods on web applications.

ЗМІСТ

ВСТУП

У наш час стало актуально зберігати інформацію в електронному вигляді, яка знаходиться на веб-серверах. Таким чином важливим постає питання способів збереження цілісності та захисту веб-ресурсів, серверів та клієнтів від зловмисників (зміни або використання особистої інформації у власних цілях сторонніми особами).

Веб-сервер (web-server) – це сервер, що відповідає за прийом і обробку запитів (HTTP-запитів) від клієнтів до веб-сайту . В якості клієнтів зазвичай виступають різні веб-браузери. У відповідь веб-сервер видає клієнтам HTTPвідповіді, в більшості випадків, разом з HTML-сторінкою, яка може містити:

всілякі файли, якісь зображення, медіа-потік або будь-які інші дані.

Клієнт користувача, яким переважно є веб-браузер, передає веб-серверу запити на отримання ресурсів. Ресурси – це HTML-сторінки, цифровий медіа контент, медіа-потоки, різні зображення, файли даних, необхідні клієнту. У відповідь веб-сервер передає клієнтові запитані ним дані. [1]

В світі є величезна кількість ботнетів деякі з них існують більше 10 років поспіль. Це можна побачити виходячи з таблиці

Дата створення	Ім'я	Кількість ботів	Потенціал для спаму	Подібні ботнети
2009 (Травень)	BredoLab	30,000,000[3]	3.6 млрд./день	Oficla
2008 (приблизно)	Mariposa	12,000,000[4]	?	Немає
?	Conficker	10,000,000+[5]	10 млрд./день	DownUp, DownAndUp, DownAdUp, Kido
?	Zeus	3,600,000 (лише США)[6]	Немає	Zbot, PRG, Wsnpoem, Gorhax, Kneber
2007 (приблизно)	Cutwail	1,500,000[7]	74 млрд./день	Pandex, Mutant
?	Grum	565,000[8]	39.9 млрд./день	Tedroo
?	Kraken	495,000[9]	9 млрд./день	Kracken
2007 (Березень)	Srizbi	450,000[10]	60 млрд./день	Cbeplay, Exchanger
?	Lethic	260,000[11]	2 млрд./день	Немає
?	Mega-D	250,000[12]	10 млрд./день	Ozdok
2004 (Початок)	Bagle	230,000[11]	5.7 млрд./день	Beagle, Mitglieder, Lodeight
?	Bobax	185,000	9 млрд./день	Bobic, Oderoor, Cotmonger, Hacktool.Spammer, Kraken
?	Torpig	180,000[13]	Немає	Sinowal, Anserin
?	Storm	160,000[14]	3 млрд./день	Nuwar, Peacomm, Zhelatin
2006 (приблизно)	Rustock	150,000[15]	30 млрд./день	RKRustok, Costrat
?	Donbot	125,000[16]	0.8 млрд./день	Buzus, Bachsoy
2008 (Листопад)	Waledac	80,000[17]	1.5 млрд./день	Waled, Waledpak
?	Maazben	50,000[11]	0.5 млрд./день	Немає
?	Onewordsub	40,000[18]	1.8 млрд./день	?
?	Gheg	30,000[11]	0.24 млрд./день	Tofsee, Mondera
?	Nucrypt	20,000[18]	5 млрд./день	Loosky, Locksky
?	Wopla	20,000[18]	0.6 млрд./день	Pokier, Slogger, Cryptic
2008 (приблизно)	Asprox	15,000[19]	?	Danmec, Hydraflux
?	Spamthru	12,000[18]	0.35 млрд./день	Spam-DComServ, Covesmer, Xmiler
?	Xarvester	10,000[11]	0.15 млрд./день	Rlsloup, Pixoliz
2009 (Серпень)	Festi	?	2.25 млрд./день	Немає
2008 (приблизно)	Gumblar	?	?	Немає
?	Akbot	?	?	Немає

Таблиця 1. Найбільші ботнети світу[2]

Виходячи з статистичних даних за день відбувається більше 2200 атак, тобто в середньому це приблизно 1 раз в 39 секунд. І кожного року кількість атак лише зростає. Звертаючи увагу на цю невтішну статистику можна зробити висновок що мати надійну систему захисту критично необхідно для компаній та корпорацій будь-якого розміру.

Але мають системи захисту зазвичай компанії та інші установи, а звичайні користувачі захищені значно гірше. В першу чергу це пов’язано з відсутністю ком’ютерної грамотності у користувачів і дуже широко поширені стандартні паролі типу «12345678» «qwertyuiop» та інші. Найбільш відомі паролі наведено нижче

1. 123456

2. 123456789

3. qwerty

4. 12345678

5. 111111

6. 1234567890

7. 1234567

8. password

9. 123123

10. 987654321

11. qweryuiop

12. mynoob

13. 123321

14. 666666

15. 18atcskd2w

16. 7777777

17. 1q2w3e4r

18. 654321

19. 555555

20. 3rjs1la7qe

21. google

22. 1q2w3e4r5t

23. 123qwe

24. zxcvbnm

25. 1q2w3e

[3]

Ботнети постійно еволюціонують та покращуються та стають більш масовими ніж були. Часто сучасні ботнети використовують модульну структуру, це допомагає в режимі реального часу змінювати їх функціонал, а також приховувати свою діяльність. І вибір розробників часто падає на якість відомі сервіси як наприклад було з NightOwl. Ця программа змінювала тему на темну на Mac коли цієї функції ще не було реалізовано. Потім цю программу викупили а потім через рік знайшли шкідливий код який відправляв інформацію третім особам.

Також розглядаючи методи приховування ботнету треба виділити поліморфізм там метаморфізм. Поліморфні віруси змінюють свій код при кожному вживленні, а метаморфні можуть повністю себе переписати що робить дуже складним їх пошук для антивірусного ПО.

Також обфускація дуже заважає в їх пошуку, роблячи код незрозумілим.

Сучасні ботнети стали відходити від класичного управління(C&C) до управління через P2P-мережі, що робить їх стійкими до відключення порівняно з традиційними командними серверами.

Деякі розробники пішли далі та стали використовувати блокчейн в якості командних серверів, що робить їх майже ідеально захищеними від спроб їх відключення.

Звертаючи увагу на активний розвиток технологій з впевненістю можна сказати що в найближчому майбутньому з’являться більш сучасні та технологічні ботнети. Вже зараз з активним розвитком штучного інтелекту зловмисники використовують і його наприклад для генерації фішингових листів або якихось фейкових новин. Генеруються фейкові чеки, дані та інше і це тільки мала частина того що зловмисники роблять за допомогою ШІ.

Тож виходячи з цього потрібно звернути увагу на цю проблему і шукати ефективні методи боротьби з розповсюдженням а також з наслідками атак ботнетів.

Мета Знайти найефективнішу стратегію атаки з урахуванням захисту на прикладі fail2ban. Визначити можливості протидії такій стратегії.

1. СИСТЕМИ ЗАХИСТУ ВІД БРУТФОРСУ

1.1 Ботнети и їх організація

Брутфорс-атаки є одним із найпоширеніших методів злому облікових записів, серверів, адмін-панелей та інших додатків які мають певну систему автентифікації.

Вичерпний пошук (англ. exhaustive search) або метод грубої сили (англ. brute-force) - загальний метод розв'язання задач та алгоритмічна парадигма[en], суть якого в систематичному перенумерованні всіх можливих кандидатів на розв'язок і перевірці кандитата на виконання умов.

Для прикладу, алгоритм повного перебору для пошуку дільників натурального числа n. Перераховуємо всі натуральні числа від 1 до n, і перевіряємо кожен з них чи буде він ділити n без остачі. У задачі про вісім ферзів повний перебір полягає в тому, щоб розглянути всі можливі розташування 8 місць з 64 клітин шахової дошки, і, для кожного з них, перевірити, чи може кожна (королева) місце атакувати будь-яке інше.

Брутфорс-атаки можуть бути класифіковані наступними видами:

Проста брутфорс-атака: Зловмисник використовує автоматизацію та скрипти для угадування паролів шляхом перебору всіх можливих комбінацій.

Атака по словнику: Зловмисник використовує словник з популярними паролями для швидшого злому.

Обернена атака грубої сили: Зловмисник використовує вже відомі паролі для злому інших облікових записів.

Гібридна атака грубої сили: Комбінує методи брутфорсу та атак по словарю для ефективнішого злому.

Дублювання паролів: Зловмисник використовує паролі, які вже були зламані раніше, для злому інших облікових записів.

Ботнеты: Зловмисник використовує мережу комп'ютерів для швидшого перебору можливих комбінацій паролів.

Персональний взлом: Зловмисник використовує соціальну інженерію для створення індивідуального словника для кожного жертви, що може включати імена родичів, друзів, домашніх тварин, важливі дати та інші особисті деталі.

«Брут-чек»: Зловмисник намагається зламати паролі у великих кількостях для заволодіння даними не одного користувача, а багатьох.

Серед найпоширеніших способів брутфорсу перебор всіх символів який включає всі символи від A до Z в обох реєстрах, спеціальних символів, та цифр.

Перебор за допомогою словника наприклад за частотою використання RockYou [4]. Це один з найвідоміших словників деякі його варіації часто йдуть в певних варіаціях лінукс систем як Kali Linux. В ньому зібрані всі найбільш часто використовувані паролі в світі. Є дуже велика кількість його варіацій доповнень та правок. Його часто використовують для підбору паролів до деяких систем та веб-додатків. Прикладами найчастіше використовуваних паролів є всесвітньо відомі «12345678» «qwertyuiop» «password» «admin» та інші. Дуже часто в внутрішній інфраструктурі використовують ці стандартні паролі і зловмисник зазвичай витратить значно менше часу перебираючи стандартні паролі або найбільш часто використовувані ніж буде перебирати всі символи.

Перебор за допомогою спеціалізованих словників за певними ознаками(іменами тварин, географічними об’єктами, тощо) [5]. Більшість цих словарів сгенеровані за допомогою HashCat та деякі з них базуються на RockYou. Їх використовують коли є певна інформація про вподобання цілі або інші певні ознаки з яких можна зробити висновок про доцільність їх використання.

Перебор за допомогою словників з витоків баз різних сервісів, організацій або установ. [6] Бази скомпрометованих данних часто використовують для більш широко ураження цілі, наприклад є витік з бази певного банку з паролями та іншими данними і зловмисник за допомогою цих даних може продовжити розвідку певної цілі для наприклад крадіжки облікових записів у соцмережах(бо нажаль використання однакових паролів є дуже розповсюдженим явищем) або для того щоб отримати доступ до інших облікових записів у інших банках. Також ці бази використовують для поповнення бази найчастіше використовуваних паролів або генерації своєї бази заснованої на цих даних.

1.2 Захист від брутфорсу

Fail2ban - це безкоштовна популярна програмна платформа з відкритим вихідним кодом, розроблена для запобігання вторгнень, яку можна використовувати для захисту вашого сервера від атак методом перебору паролів. Одним із головних завдань Fail2ban є блокування IP-адреси, активність якої має явну шкідливу ознаку. Принцип роботи Fail2ban відстежує файли журналів (наприклад, /var/log/auth.log, /var/log/apache/access.log), тимчасово або постійно забороняючи підозрілу активність IP-адреси шляхом оновлення наявних правил брандмауера. Fail2ban дає змогу автоматично виконати різні дії, як-от заблокувати IP, використовуючи правила iptables, або просто надіслати повідомлення електронною поштою адміністратору сервера.

За замовчуванням Fail2ban постачається з правилами фільтра під різні сервіси (sshd, apache, mysql proftpd, тощо), але конфігурацію можна легко розширити для моніторингу будь-якої іншої служби. Усі фільтри та дії налаштовуються у файлах конфігурації, тож Fail2ban є чудовим гнучким інструментом для запобігання злому вашого сервера. Після досягнення певної кількості невдалих спроб, Fail2Ban автоматично блокує відповідні IP-

адреси, запобігаючи подальшим спробам вторгнення. Основні переваги Fail2Ban включають:

1 Автоматизація безпеки: Fail2Ban дає змогу автоматично реагувати на загрози без необхідності втручання адміністратора.

2. Налаштування гнучких правил: Адміністратори можуть налаштовувати правила для моніторингу різних служб і журналів відповідно до конкретних потреб.

3. Зниження навантаження: Блокування зловмисних IP-адрес допомагає знизити навантаження на сервер, оскільки атаки невдалих спроб аутентифікації припиняються на ранній стадії.

4. Легка інтеграція: Fail2Ban легко інтегрується з іншими інструментами безпеки та системами моніторингу. Використання Fail2Ban є важливим елементом забезпечення безпеки серверів і ресурсів у мережі. Цей інструмент допомагає поліпшити захист від зовнішніх загроз і знизити ризик успішних атак на вашу систему.

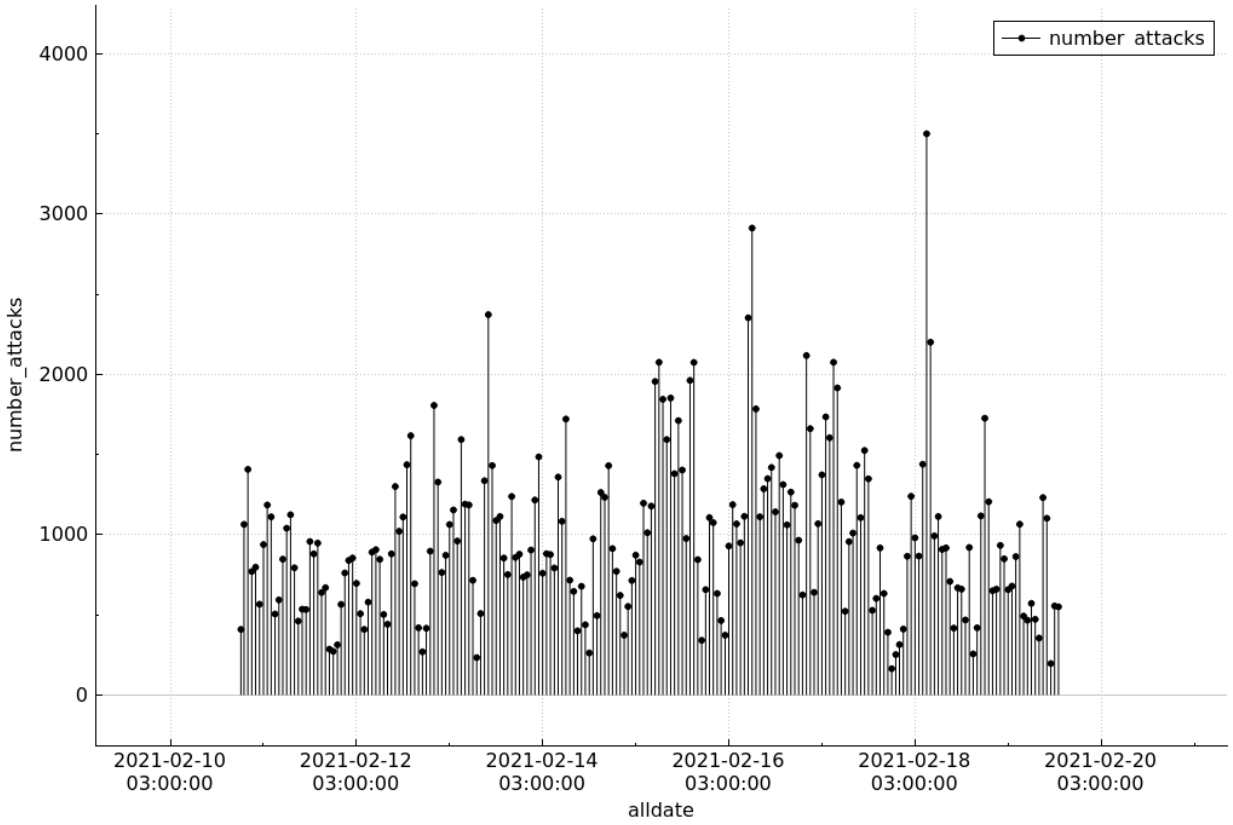
Принцип роботи fail2ban оснований на блокуванні на певний час(далі T) після N спроб. Це означає що після певної кількості невдалих спроб буде заблокована можливість на певний час підбирати паролі. Наприклад ip-адреса 123.123.123.123 намагається підключитись до адмін панелі сайту site.com, в налаштуваннях fail2ban зазначена максимальна кількість спроб 5 і після цього він блокує цей ip на 10 хвилин. Це є дефолтними налаштуваннями fail2ban які можна змінити в файлі конфігурації. Файл налаштувань можна знайти за стандартним шляхом /etc/fail2ban/jail.conf.

1.3 Можливості протидії fail2ban

Бот-ферми роблять більш гнучкими та часто налаштовують їх з урахуванням певних программ захисту. Наприклад включаючи стандартні налаштування захисних програм в моделі атаки. Дуже часто використовуються стандартні

налаштування в программах захисту і в випадку з fail2ban зловмисник може використовувати таку схему атаки

Наприклад зловмисник знаючи стандартні налаштування fail2ban а саме 5 спроб і бан на 10 хвилин може коригувати дії своїх ботів наприклад робити 4 спроби і чекати 10 хвилин після останньої спроби до наступного циклу щоб уникнути блокування ip-адреси.



Графік 1.1 Атаки на сервер

2. СТРАТЕГІЯ ВИЗНАЧЕННЯ ПАРАМЕТРІВ СИСТЕМ ЗАХИСТУ

2.1 Проблема підлаштування під параметри захисту від брутфорсу

В світі існує велика кількість працюючих ботнетів які щосекунди здійснюють різноманітні атаки на корпорації, державні установи, адмін панелі сайтів або облікові записи звичайних користувачів. Ботнет може існувати та функціонувати по різному, в ньому може бути якась певна кількість ботів (уражених вірусом ком'ютерів які виконують певні дії) наприклад намагаються DdoS'ити певні веб додатки або займаються підбором паролів до цілей які вказані у їх програмі. Також зв'язок між ботнетами може бути побудований у різних форматах. Бот може мати зв'язок з панеллю управління та\або з іншими ботами. Він має зв'язок у певних проміжках часу або вони встановлюють зв'язок коли з'являється нова вказівка або зміна попередньої або вони тримають зв'язок весь час.Також можлива ситуація коли боти можуть бути пов'язані один з одним та передавати один одному певну інформацію . Бот працює за раніше вкладеною програмою. Бот ще при зараженні отримує певну інформацію щодо цілей та працює самостійно поки не виконає задачу

Ціль стратегії максимізувати кількість спроб доступу до системи при підборі паролю протягом певного проміжку часу

2.2 Стратегія поведінки для роботи рою

Система рахує кількість невдалих спроб з одної адреси і якщо воно перевищує ліміт за певний проміжок часу то тимчасово обмежується доступ цій адресі на певний проміжок часу

Розлянути захист можна на прикладі fail2ban він банить на 10 хвилин після 5 невдалих спроб за 10 хвилин T_{BAN} , кількість спроб за N_{AT} після якого наступає бан, час на 1 спробу T_{AT} та час за який робиш певну кількість спроб та наступає бан $T_{DELTA.}$ З цього слідує що якщо час бану більше за час за який потрібно зробити спроби если ban time >> delta T то треба робити на 1 спробу менше за відведений час N - 1 за delta T

Можливості обходу fail2ban для ботнету

Є дві стратегії атаки для ботнету

Паралельна

При паралельній атаці є дуже великий ризик покласти сервер бо сервер за короткий час буде отримувати дуже багато запитів і можливо не зможе їх обробити. Також великою проблемою буде те що на цю активність зверне увагу адміністратор ресурсу або служба безпеки організації. Але в той же час це найвидший шлях щоб підібрати пароль методом брутфорсу.

Послідовна

При послідовній атаці ризику покласти сервер майже немає але є інша проблема це час. При послідовній атаці всі боти послідовно будуть підбирати пароль і це буде займати значно більше часу ніж при паралельній. При цьому ризик того що адміністратор чи служба безпеки зверне увагу на підозрілу активність не знижується.

Ігнорування

Суть цього методу робити максимальну кількість спроб не зважаючи на реакцію захисту та встановлені методи захисту. Тепер необхідно зрозуміти скільки спроб буде зроблено за певний час. Для цього методу можна навести формулу $N = (T / (T_{BAN} + N_{AT} * T_{AT})) * N_{AT} * K$

У разі якщо $T_{AT} << T_{DELTA}$ то $N = (T / T_{BAN}) * N_{AT} * K$

Формула показує максимум що можна зробити виходячи з цієї стратегії при атаці за допомогою координованої атаки.

Підлаштування

Стратегія підлаштування під стандартні налаштування реалізується в випадку с програмою захисту fail2ban треба робити на 1 спробу менше $N_{AT} - 1$

і чекати поки не закінчиться час бану. Це можна навести формулою $N = (T / (T_{DELTA} + (N_{AT} - 1) * T_{AT}) * (N_{AT} - 1)$

Проблеми про обході fail2ban

Серед основних проблем обходу схожих за принципом дії на fail2ban захисних систем це його налаштування. Налаштування можна дізнатися лише під час самої атаки на етапі розвідки цілі використавши якогось бота для цього. Для того щоб дізнатися час на спроби та кількість самих спроб необхідно спробувати ввести максимальну кількість спроб а коли заблокують продовжувати намагатися послідовно підібрати пароль з інтервалами наприклад 10 секунд по черзі кожним ботом, тобто кожен наступний бот буде з більшим на 10 секунд інтервалом ніж попередній поки у якогось бота не вдасться продовжити спроби.

Стратегія ігнорування не дає можливості найбільш ефективно використовувати наявні ресурси а також є шанс того що це може покласти сервер, від чого пропаде можливість продовжувати атаку. Також дуже велика ймовірність того що адміністратор ресурсу зверне увагу на підозрілу активність та переадресує атаку на honeypot або заблокує можливість авторизації від чого атака зупиниться. Ще одною проблемою являється неефективність атаки, ця проблема витікає з назви стратегії – ігнорування.

Ігноруючи методи захисту не можна зробити атаку ефективною. Актуальність цієї стратегії може бути лише у разі атаки на потужні системи у разі цілі підібрати дані методом брутфорсу, ця стратегія краще підходить у якості DdoS-атаки.

Підлаштування під стандартні налаштування

З очевидних переваг цієї стратегії це її простота. Треба лише 1 раз задати налаштування для ботнету та вони будуть просто їх повторювати. Це не потребує уваги з боку власника бот-ферми, та постійних змін налаштувань,

також виходячи з цього бот ферма не обов’язково потрібна мати зв’язок з панеллю керування та іншими ботами.

З мінусів цієї стратегії варто зазначити що вона не буду підходити до великого об’єму серверів. Значна кількість серверів мають свої налаштування які будуть відрізнятись від стандартних. Також це буде знижувати ефективність атаки враховуючи відмінності від стандартних налаштувань, наприклад на сервері налаштування на максимум 3 спроби за 20 хвилин, в той час як на бот фермі використовують стандартні налаштування на 5 спроб за 10 хвилин тож ця атака буде марно витрачати ресурси та втрачати свою ефективність, крім того звертати увагу адміністратора на збільшення кількості заблокованих адрес.

Ця стратегія буде актуальна у разі якщо ми не будемо мати зв’язку з паннеллю керування або з іншими ботами або якщо мати велику кількість ботів та часу. Підлаштування під налаштування певної системи захисту

У цієї стратегії є 2 варіанта яким чином це можна зробити:

У першому варіанті ми спочатку методом перебору намагаємось знайти ці налаштування. З плюсів це більше простий варіант який не потребує уваги адміністратора бот-ферми. Система як знайде найкращий варіант почне по ньому працювати без втручання людини. З очевидних мінусів це потребує великого об’єму часу та ресурсів бо перебираючи по черзі всі варіанти буде заблоковано багато адрес та буде можливість у адміністратора звернути увагу на підозрілу активність. Цей варіант буде актуальним при тимчасовому або постійному зв’язку та при значній кількості задіяних у бот-фермі адрес.

Розглядаючи другий варіант використовуючи метод бінарного пошуку ми можемо зазначити з плюсів те що це найбільш ефективний алгоритм тому що буде орієнтований під налаштування конкретної цілі. Це дає можливість підлаштуватися та найбільш ефективно підбирати дані для авторизації.

З мінусів можна зазначити складність цього способу, бо це потребує значних знань щоб написати алгоритм який буде самостійно шукати та розуміти коли він знайде межі налаштувань а також скоординувати атаку інших ботів, також це потребує постійного зв’язку для координації інших ботів та можливості динамічно змінювати їх налаштування.

Отже цей варіант не підійде у випадку якщо у власника бот-ферми або программи нема постійного зв’язку з іншими ботами, бо вони не зможуть координувати атаку, також це потребує створення самого алгоритму. Проте цей варіант буде актуальним якщо потрібно максимально швидко та точно підлаштуватися під певну ціль або маючи обмежену кількість ботів.

2.3 Можливість визначення параметрів шляхом аналізу спроб з’єднання

Вибір оптимальної стратегії для ботів залежить від параметрів N_{AT} та T_{BAN}

Максимальну кількість N_{AT} бана визначити дуже легко за допомогою наведених нижче методів

За проміжком часу необхідному на з’єднання. Парадоксально але в випадку бану час на з’єднання буде значно менше ніж у випадку неправильного паролю. Сервер витрачає найбільше часу на перше з’єднання бо він кешує з’єднання та супровідні дані а після цього вже перевіряє саме з’єднання на правильність та на відсутність у списку заблокованих адрес. Сервер приймаючи пакети на авторизацію крім логіну та паролю дає певний час користувачу на від’єднання або для отримання додаткових параметрів. В випадку бану час з’єднання зазвичай менше секунди.

За сповіщенням протоколів. В нашому випадку ми будемо використовувати бібліотеку paramiko для отримання протоколів і звичайно використовувати термінологію цієї программи. Спираючись на її сповіщення ми будемо розуміти що саме відбувається під час спроб підключення до серверу.

Отримавши помилку AuthenticationException ми розуміємо що сталась невдала авторизація. Отже fail2ban ще не вніс на в список заблокованих та ми

можемо спробувати ще раз підключитись. У випадку отримання помилки NoValidConnectionsError ми розуміємо що користувач вже занесений до списку заблокованих адрес і залишається лише чекати певний час до розблокування. Цей метод є більше точний у порівнянні з визначенням по часу з'єднання бо ми завжди розуміємо що саме відбувається з сервером. Але хоч якими ефективними ці методи не були нам треба визначити час бану оскільки при повторних спробах з'єднання при бані час бану буде подовжуватися з моменту останньої спроби. Тому використавши всі спроби і не знаючи часу бану продовжувати постійно намагатися з'єднатися буде безрезультатно бо ми будемо отримувати помилку NoValidConnectionsError.

2.4 Стратегії визначення часу бану

Визначення часу бана повинно виконуватися ботнетом шляхом послідовних спроб. Кожний з ботів вибирає якесь значення часу T_N через яке спробує з'єднатися ще раз. Перше значення T_N вибирається найближчим до стандартних налаштувань. В процесі підбору результат неважливо який позитивний чи негативний сповіщується всьому ботнету щоб не було повторних спроб тих самих значень. Наступний бот виконує ту ж процедуру, але вже з урахуванням минулих результатів використовуючи інше значення в залежності від результату попереднього.

Є кілька стратегій які ми розлянемо

Перша стратегія полягає у послідовному пошуку або ж випадковим шляхом підбираючи значення. Наприклад вибрати інтервал в 10 секунд і кожним наступним ботом збільшувати значення на цей інтервал поки не буде досягнуто результату. Таким чином час підбирається лінійно до досягнення позитивного результату, також важливо буде перевірити значення трохи менше за результат першого позитивного результату бо крок інтервалу може бути занадто великий і буде втрачено певну кількість зайвого часу і це буде негативно впливати на ефективність брутфорсу.

Друга стратегія використовує метод бінарного пошуку. Цей метод має схожість з попереднім але є більш ефективним бо буде задіяна менша кількість спроб ніж у разі з послідовним підбором. Суть методу полягає у отриманні будь якого позитивного результату, тобто помилки AuthenticationException і в подальшому зменшення значення у 2 рази.

Наприклад налаштуваннями fail2ban є 5 спроб за 24 хвилин, бот бере великий інтервал наприклад 10 хвилин і використавши лише 3 боти ми розуміємо що інтервал буде менше або дорівнює 30 хвилинам. Потім зменшуємо інтервал в 2 рази тобто бот чекає 25 хвилин і знову отримає позитивний результат потім ще навпіл і так до того часу поки ми не знайдемо необхідне значення з тою точністю до якої нам необхідно, наприклад в данному випадку мілісекунди нас не цікавлять а секунди повністю влаштовують. Таким чином ми вираховуємо час за $\log_2$. Ця стратегія буде найбільш економна з точки зору витрати ресурсів та найбільш швидка у порівнянні з минулим способом, бо зазвичай інтервали на більш ніж годину використовують дуже рідко. І в результаті отримавши точний час ми можемо зробити нашу атаку настільки ефективною наскільки це можливо виходячи з можливостей наявних ресурсів.

3. ТЕСТУВАННЯ СТРАТЕГІЙ ВИЗНАЧЕННЯ ПАРАМЕТРІВ СИСТЕМ ЗАХИСТУ

3.1 Система тестування

Для тестування буде використано программу для віртуалізації virtualbox.

Для системи буде використано 3 мережеві карти .

В віртульній машині створюємо 2 мережеві адаптери

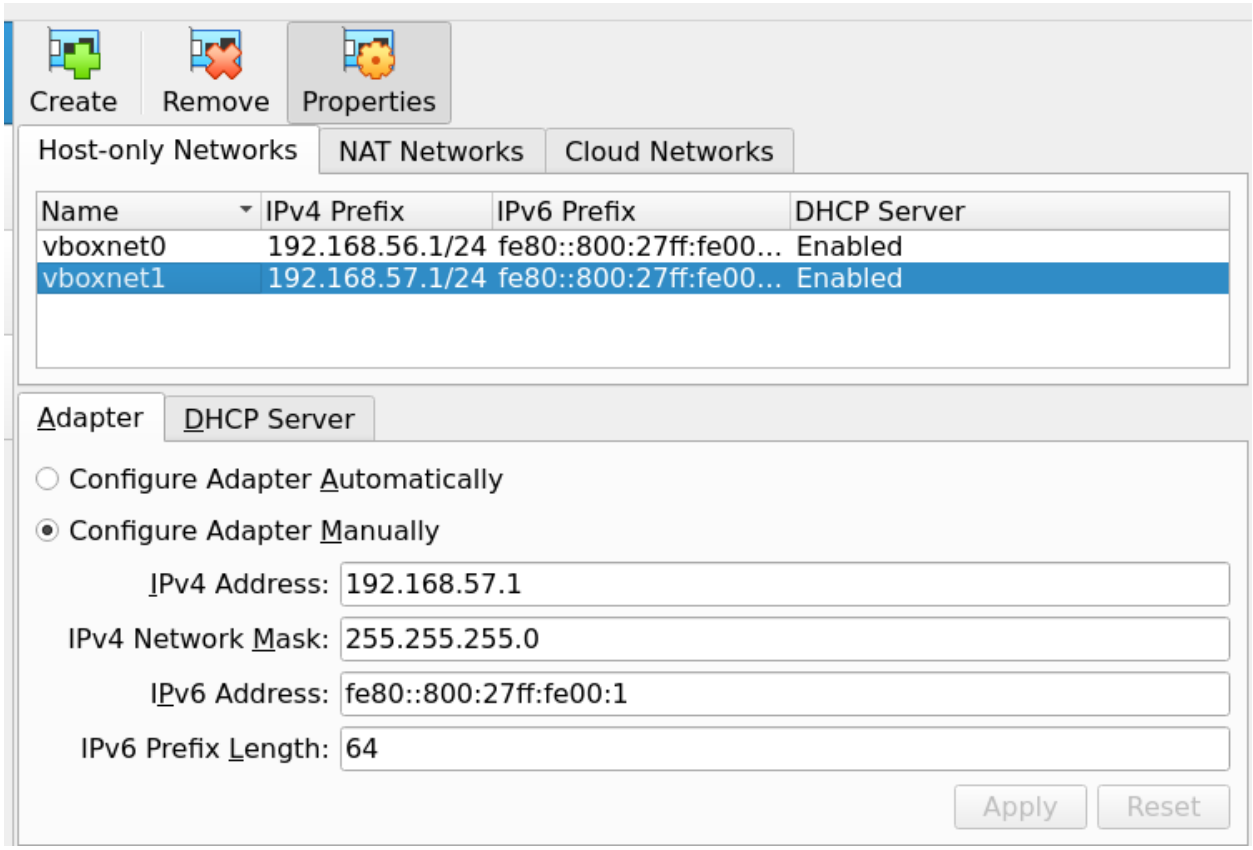


Рисунок 3.1 - Мережеві адаптери

Перший адаптер буде відповідати за інтернет NAT

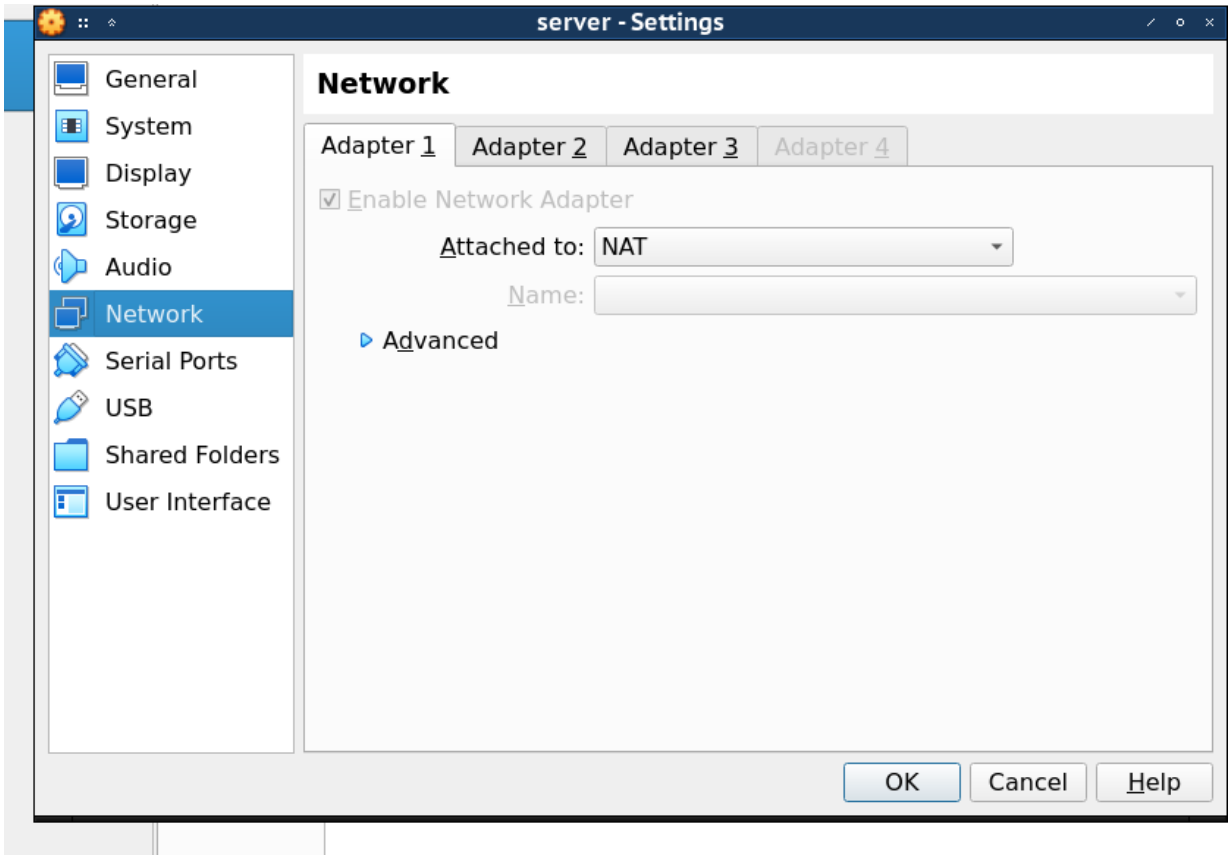


Рисунок 3.2 - NAT

Другий адаптер це керуючий інтерфейс з доступом по ssh

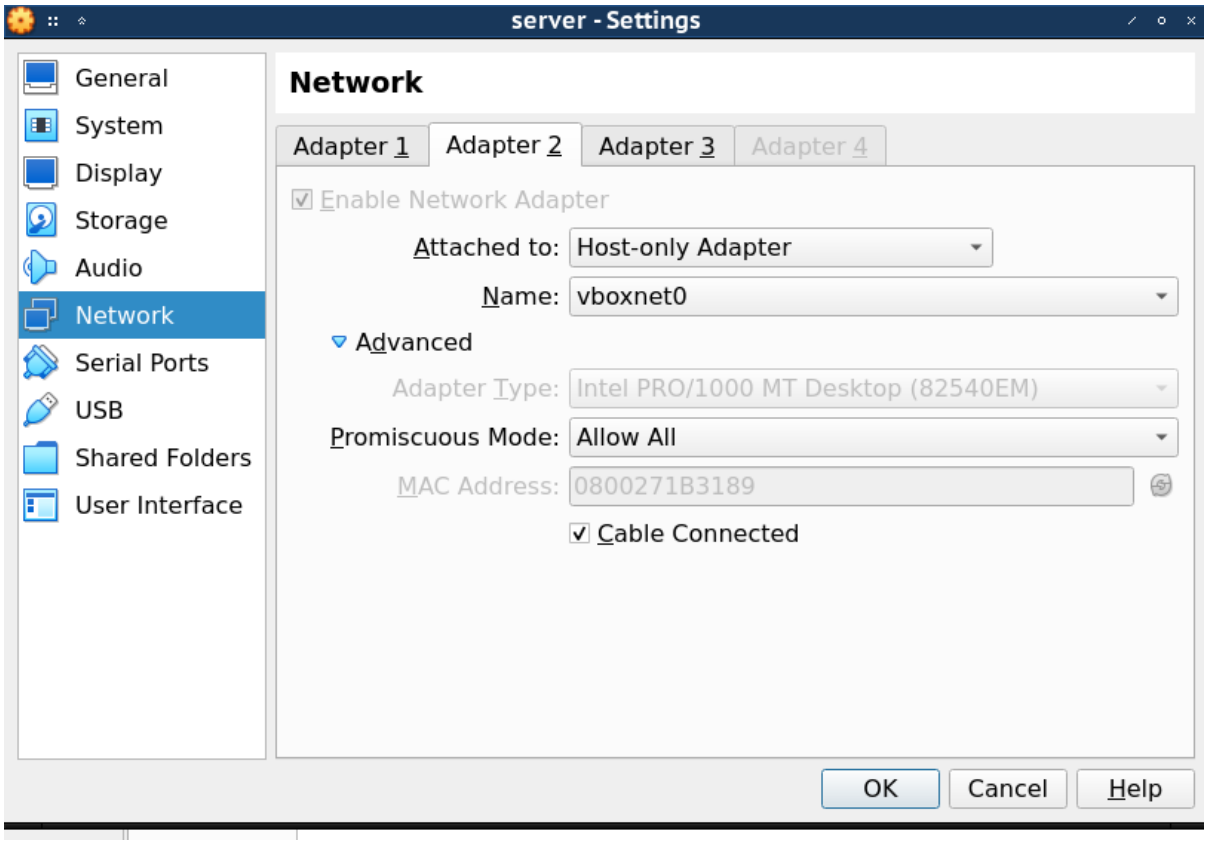


Рисунок 3.3 – Доступ по ssh

Третій адаптер це тестовий інтерфейс для моделювання атак

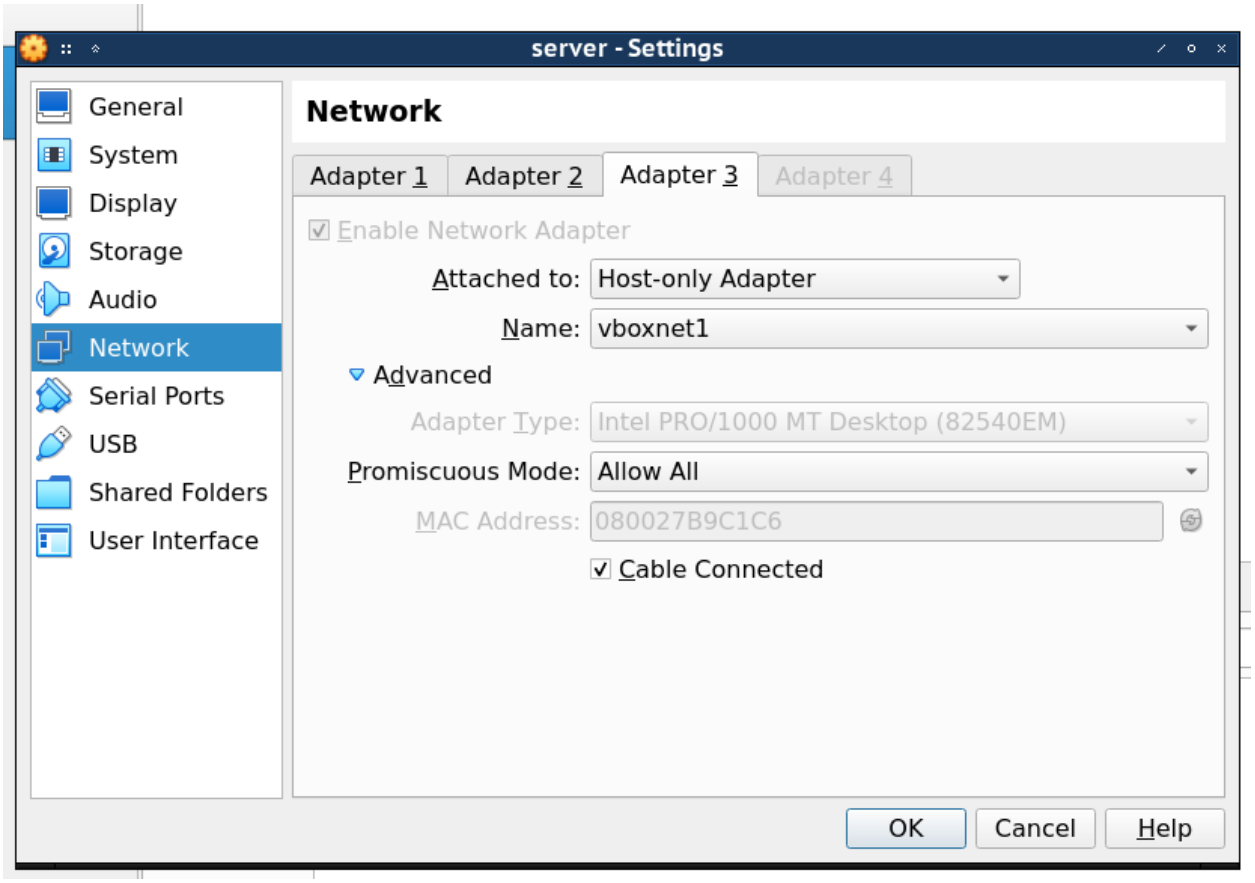


Рисунок 3.4 – Атакуюча машина

В якості віртуальної машини буде виступати машина з встановленою Debian а саме Linux 5.10.0-28-amd64 #1 SMP Debian 5.10.209-2 (2024-01-31) x86_64 GNU/Linux

Також нам знадобиться певне програмне забезпечення а саме OpenBSD ssh server та fail2ban 0.11 2-2

```
mini@mini:~$ systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-05-18 18:28:20 EEST; 42min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 306 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 344 (sshd)
      Tasks: 1 (limit: 11854)
     Memory: 8.5M
        CPU: 441ms
     CGroup: /system.slice/ssh.service
             └─344 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
```

Рисунок 3.5 – Демонстрація роботи OpenBSD ssh service

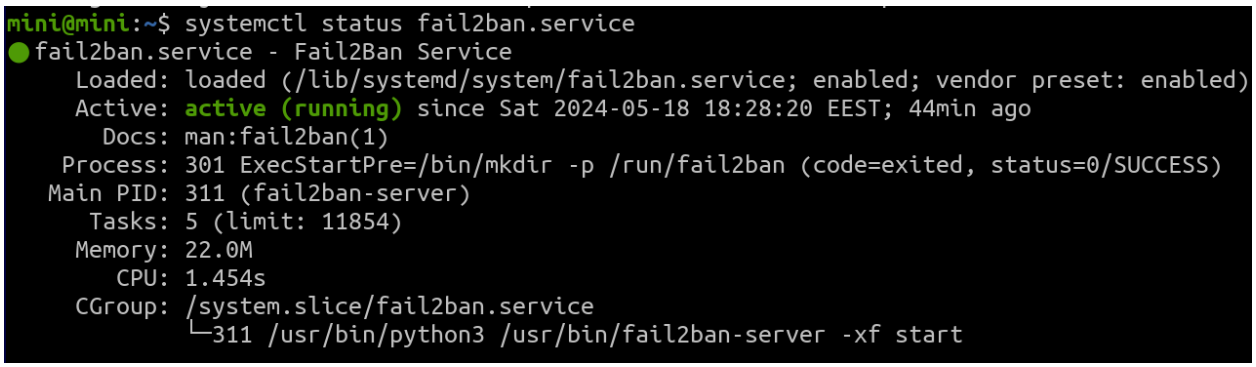


Рисунок 3.6 – Демонстрація роботи fail2ban

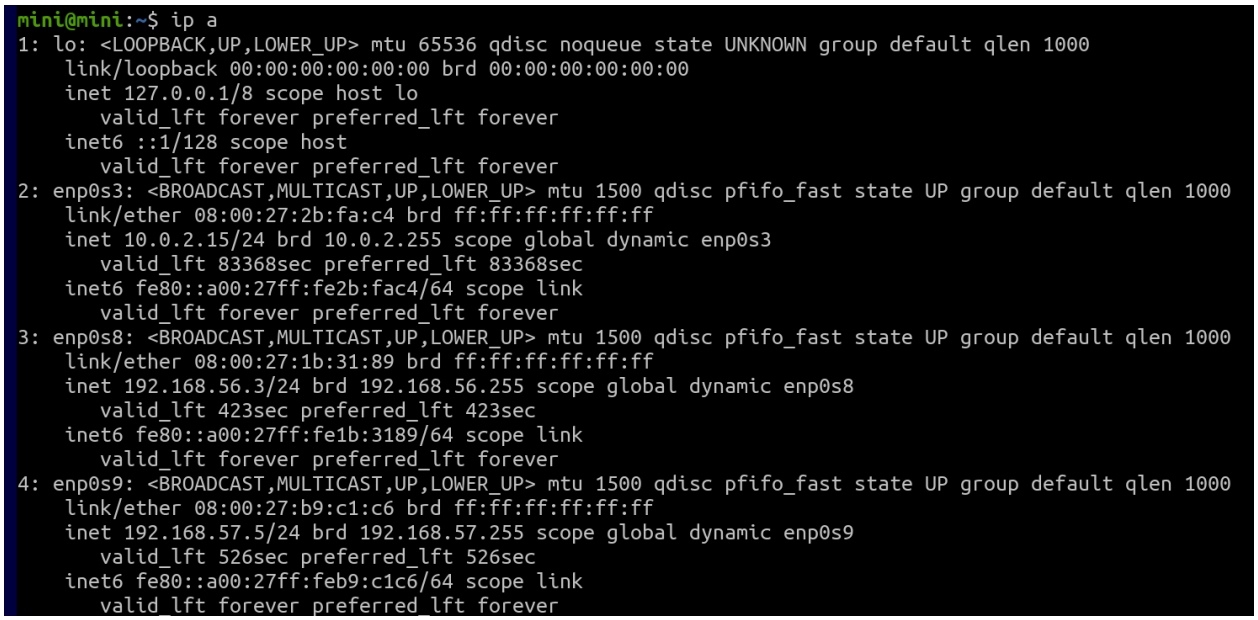


Рисунок 3.7 – Демонстрація мережевих карт: enp0s3 – NAT, enp0s8 - керуючий адаптер, enp0s9 - тестовий інтерфейс

3.2 Робота fail2ban

Ми будемо моделювати атаку на прикладі ssh-серверу як найбільш частої цілі. Аналіз кількості спроб здійснюється за допомогою /var/log/auth.log

Рисунок 3.8 Приклад атаки на сервер

Роботу системи можна контролювати за допомогою fail2ban client

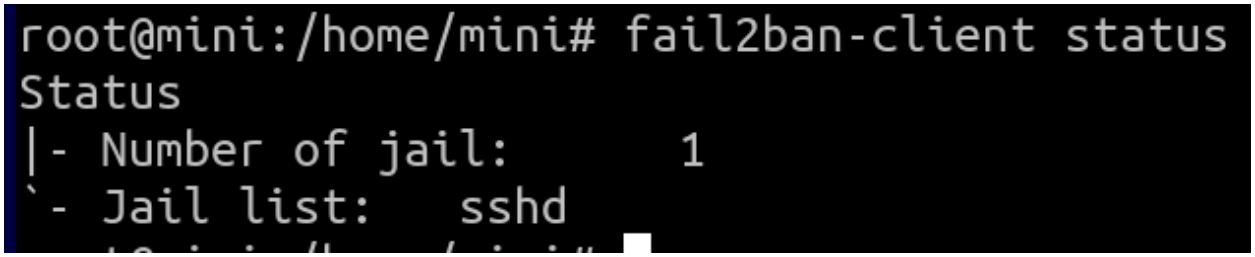


Рисунок 3.9 - Загальний статус

Рисунок 3.10 - Статус певного сервісу

Рисунок 3.11 - Демонстрація перезавантаження за запуску з нуля

3.3 Тестуючий сканер

В якості скрипта для тестування використовується скрипт на мові програмування python з використання бібліотеки paramiko [7]

Ця бібліотека дуже проста та примітивна але для цього завдання підійде найкраще. Для більш складних операцій є бібліотека fabric.

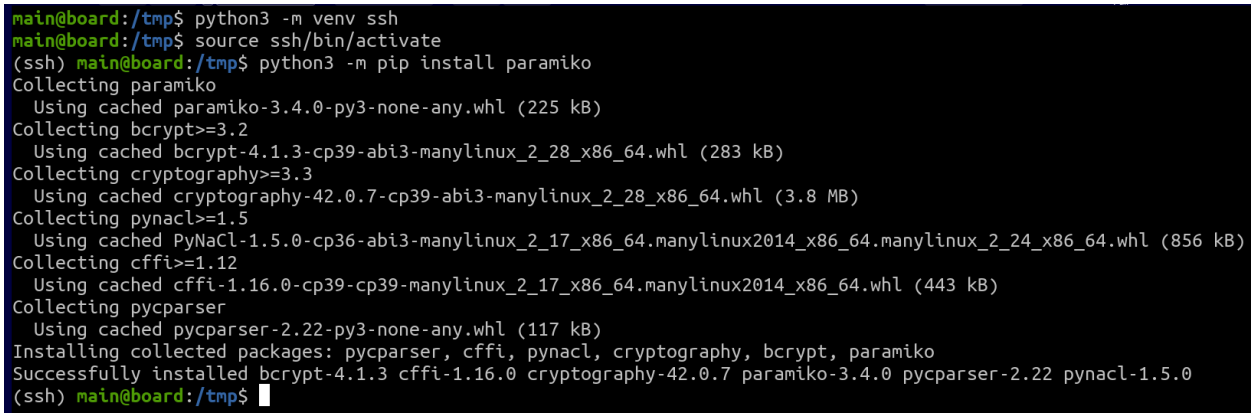


Рисунок 3.12 - Система використовує venv з заздалегідь встановленою бібліотекою

Демонстрація роботи

```
(ssh) main@board:/tmp$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import paramiko
>>> client = paramiko.SSHClient()
>>> client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
>>> hostname = '192.168.57.5'
>>> username = 'mini'
>>> password = '1'
>>> port = 22
>>> client.connect(hostname, port, username, password)
>>>
```

Рисунок 3.13 - З’єднання з сервером

```
May 18 19:45:55 mini sshd[603]: Accepted password for mini from 192.168.57.1 port 59798 ssh2
May 18 19:45:55 mini sshd[603]: pam_unix(sshd:session): session opened for user mini(uid=1000) by (uid=0)
May 18 19:45:56 mini systemd-logind[311]: New session 5 of user mini.
```

Рисунок 3.14 - Відображення з’єднання на сервері

```
(ssh) main@board:/tmp$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import paramiko
>>> client = paramiko.SSHClient()
>>> client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
>>> hostname = '192.168.57.5'
>>> username = 'mini'
>>> password = '1'
>>> port = 22
>>> client.connect(hostname, port, username, password)
>>> client.close()
```

Рисунок 3.15 - Завершення сесії

```
May 18 19:45:55 mini sshd[603]: Accepted password for mini from 192.168.57.1 port 59798 ssh2
May 18 19:45:55 mini sshd[603]: pam_unix(sshd:session): session opened for user mini(uid=1000) by (uid=0)
May 18 19:45:56 mini systemd-logind[311]: New session 5 of user mini.
May 18 19:49:46 mini sshd[603]: pam_unix(sshd:session): session closed for user mini
May 18 19:49:46 mini systemd-logind[311]: Session 5 logged out. Waiting for processes to exit.
May 18 19:49:46 mini systemd-logind[311]: Removed session 5.
```

Рисунок 3.16 - Закриття сесії на сервері

```
>>> username = 'min'
>>> client.connect(hostname, port, username, password)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/tmp/ssh/lib/python3.9/site-packages/paramiko/client.py", line 485, in connect
    self._auth(
  File "/tmp/ssh/lib/python3.9/site-packages/paramiko/client.py", line 818, in _auth
    raise saved_exception
  File "/tmp/ssh/lib/python3.9/site-packages/paramiko/client.py", line 805, in _auth
    self._transport.auth_password(username, password)
  File "/tmp/ssh/lib/python3.9/site-packages/paramiko/transport.py", line 1603, in auth_password
    return self.auth_handler.wait_for_response(my_event)
  File "/tmp/ssh/lib/python3.9/site-packages/paramiko/auth_handler.py", line 263, in wait_for_response
    raise e
paramiko.ssh_exception.AuthenticationException: Authentication failed.
>>> client.close()
```

Рисунок 3.17 - Приклад роботи з неіснуючим користувачем

```
May 18 19:51:43 mini sshd[627]: Invalid user min from 192.168.57.1 port 55498
May 18 19:51:43 mini sshd[627]: pam_unix(sshd:auth): check pass; user unknown
May 18 19:51:43 mini sshd[627]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.57.1
May 18 19:51:45 mini sshd[627]: Failed password for invalid user min from 192.168.57.1 port 55498 ssh2
May 18 19:51:51 mini sshd[627]: Connection closed by invalid user min 192.168.57.1 port 55498 [preauth]
```

Рисунок 3.18 - Реакція серверу на неіснуючого користувача

Для розуміння відповідей сервера ми використовуємо статуси paramiko.

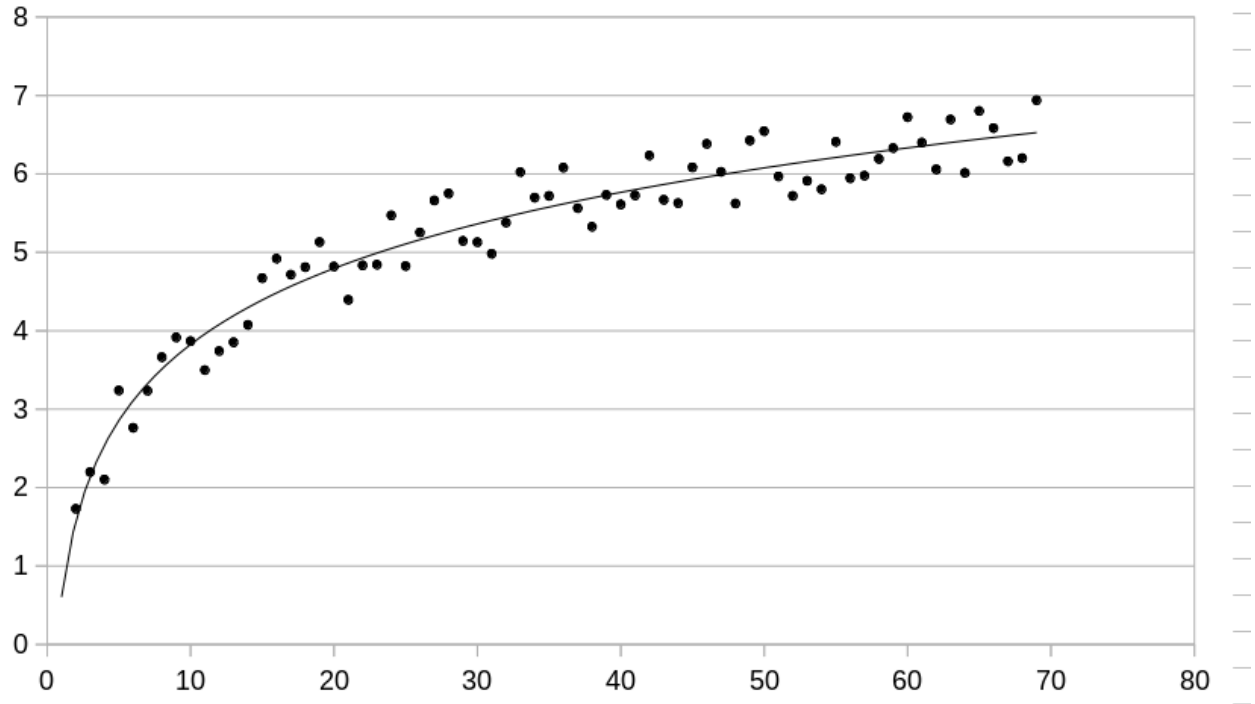


Рисунок 3.20 - Ця таблиця показує графік де шкала Y показує кроки до підбору а шкала X точність підбору

ВИСНОВОК

Кваліфікаційна робота містить дослідження методів атаки та захисту на сервери за допомогою моделювання, та запропоновано метод атаки на основі алгоритму бінарного пошуку. Досліджено стратегії атаки на сервери за використанням ботнетів для брутфорсу облікових даних для доступу по протоколу ssh. Виявлено обмеження у використанні ботнетів та було проведено аналіз стратегій атак на сервери, і пропонується використання методу бінарного пошуку для підвищення ефективності атак на сервери.

Ключовою вимогою для рішення проблеми є використання алгоритму бінарного пошуку для побудови найбільш ефективної стратегії атак на сервери з захистом схожим за принципом на модель захисту fail2ban.

У ході дослідження були вивчені теоретичні основи та методи використання ботнетів для брутфосу та захисту від них а саме статистичні дані найчастіше використовуваних паролів, принципів побудови та управління ботнетами, та стратегії атак з використанням ботнетів.

У результаті було відтворено модель атаки на сервер використовуючи різні стратегії та сформовано статистичні дані на основі проведених атак щодо ефективності методу бінарного пошуку для брутфорсу ssh-серверів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Яковенко О. П. Захист веб-ресурсів: захист веб-сервера і клієнта

Науковий керівник: к.т.н., доцент кафедри інформаційних технологій, економіки та менеджменту Москаленко А.О. URL: http://dspace.puet.edu.ua/bitstream/123456789/4411/1/%D0%97%D0%B1%D1%96%D1%80%D0%BD%D0%B8%D0%BA%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D1%96%D1%97%202016.pdf#page=112

2. Ботнет. URL: https://uk.wikipedia.org/wiki/%D0%91%D0%BE%D1%82%D0%BD%D0%B5%D1%82
3. Список найпопулярніших паролів URL: https://www.trans.eu/ua/blog/najpopuljarnishi-paroli-v-sviti-chi-vi-koristuetes-odnim-z-nih/
4. RockYou URL: https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt
5. List of password URL: https://www.openwall.com/passwords/wordlists/
6. База витоків URL: https://github.com/ohmybahgosh/RockYou2021.txt
7. Paramiko https://docs.paramiko.org
8. Статуси paramiko URL: https://docs.paramiko.org/en/3.4/api/ssh_exception.html

ДОДАТКИ

Додаток А. scanner2.py

```python
import sys
sys.path.append('/tmp/ssh/lib/python3.9/site-packages/')

import paramiko

def test_connection(client):
    try:
        client.connect(hostname, port, username, password)
        return "OK"

    # if client banned
    except paramiko.ssh_exception.NoValidConnectionsError:
        client.close()
        return "NoValidConnection"

    # if client makes wrong login
    except paramiko.ssh_exception.AuthenticationException:
        client.close()
        return "AuthenticationException"


client = paramiko.SSHClient()

# accept all keys
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

# parameters of server
hostname = '192.168.57.5'
port = 22
username = 'mini'
password = '1'


n = 0
# only 3 attempts in row
for i in range(3):
    print( test_connection(client))
```

Рисунок А.1 – scanner2.py тест

Додаток Б. scanner3.py

```python
import sys
sys.path.append('/tmp/ssh/lib/python3.9/site-packages/')

import paramiko

def test_connection(client):
    try:
        client.connect(hostname, port, username, password)
        return "OK"

    # if client banned
    except paramiko.ssh_exception.NoValidConnectionsError:
        client.close()
        return "NoValidConnection"

    # if client makes wrong login
    except paramiko.ssh_exception.AuthenticationException:
        client.close()
        return "AuthenticationException"


client = paramiko.SSHClient()

# accept all keys
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

# parameters of server
hostname = '192.168.57.5'
port = 22
username = 'min'
password = '1'


n = 0
# only 3 attempts in row
for i in range(3):
    print( test_connection(client))
```

Рисунок Б.1 тест з неправильним логіном

Додаток В. Тест з таймером

```python
import time
import functools
import sys

sys.path.append('/tmp/ssh/lib/python3.9/site-packages/')

import paramiko

def timer(func):
    """Print the runtime of the decorated function"""
    @functools.wraps(func)
    def wrapper_timer(*args, **kwargs):
        start_time = time.perf_counter()
        value = func(*args, **kwargs)
        end_time = time.perf_counter()
        run_time = end_time - start_time
        print(f"Finished {func.__name__}() in {run_time:.4f} secs")
        return value
    return wrapper_timer

@timer
def test_connection(client):
    try:
        client.connect(hostname, port, username, password)
        return "OK"

    except paramiko.ssh_exception.NoValidConnectionsError:
        client.close()
        return "NoValidConnection"

    except paramiko.ssh_exception.AuthenticationException:
        client.close()
        return "AuthenticationException"


client = paramiko.SSHClient()
# accept all keys
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

hostname = '192.168.57.5'
port = 22
username = 'min'
password = '1'


n = 0

for i in range(3):
    print( test_connection(client))
```

Рисунок В.3 – Тест показує час на з’єднання