Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій

УДК 004.75

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: "Інструментарій оцінки швидкості каналів в мережі TOR"

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ – 30.00.00.000

Студент
ІПЗ-41 Юрій РУДЕНКО

Науковий керівник
д.т.н. с.н.с. доцент Геннадій ПОРЄВ

Консультант
з питань нормоконтролю
фахівець Тамара ЧАПОВСЬКА

Допускається до захисту
Завідувач кафедри
д.т.н., професор Олексій БИЧКОВ

Київ – 2021

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ:

Завідувач кафедри
Програмних систем і технологій
_____ (Олексій БИЧКОВ)
«___»_____ 20___р.

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

_____Руденко Юрій Олегович_____
(прізвище, ім’я, по батькові)

1. Тема випускної кваліфікаційної бакалаврської роботи «Інструментарій оцінки швидкості каналів в мережі TOR» затверджена наказом вищого навчального закладу від «__»_____20__р. №_____

2. Строк здачі студентом закінченої роботи _16 червня 2021_____
3. Вихідні дані до роботи _____
_____
_____
_____
_____
4. Зміст пояснювальної записки (перелік питань, що їх належить розробити)

Дослідити принципи цибулевої маршрутизації та механізми системи Tor

Виміряти швидкість передачі даних та пришвидшити роботу цибулевої маршрутизації._____

_____

5. Перелік графічного матеріалу (з точним забезпеченням обов’язкових креслень)
_____
_____

_____

_____

_____

_____

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____ Геннадій Порєв
(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ Юрій РУДЕНКО
(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

Номер і назва етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка

1. Вивчення завдання	18.04.2021	виконано
2. Підбір та вивчення літератури	25.04.2021	виконано
3. Розробка алгоритму	05.05.2021	виконано
4. Розробка програмного продукту	08.05.2021	виконано
5. Опис розробки програмного продукту	12.05.2021	виконано
6. Підготовка презентації	05.06.2021	виконано
7. Затвердження пояснювальної записки роботи керівником	06.06.2021	виконано

Студент – бакалавр _____ Юрій РУДЕНКО

(підпис) (розшифровка підпису)

Керівник роботи _____ Геннадій Порєв

(підпис) (розшифровка підпису)

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота має обсяг 60 сторінок, містить 10 рисунків, а також 45 бібліографічних джерел.

Проблему анонімності користувача вирішують криптографічні мережі. які намагаються запобігти аналізу трафіку, змішуючи повідомлення, що проходять через мережу, таким чином, щоб зовнішній спостерігач не зміг відстежити джерело і пункт призначення повідомлення. Але, не дивлячись на те, що такі мережі працюють не один десяток років, їх швидкість усе ще залишається невисокою. У даній роботі поставлена мета проаналізувати швидкість найпопулярнішої мережі Tor та визначити чинники, що на неї впливають.

Актуальність роботи. Тема є актуальною і на сьогоднішній день, адже анонімна мережа налічує велику кількість користувачів, що вимагають стабільної та швидкої роботи.

Мета і завдання дослідження. Метою і завданням роботи є дослідження принципів цибулевої маршрутизації, механізмів системи Tor, вимірювання швидкості передачі даних та способів пришвидшення роботи цибулевої маршрутизації.

Об’єкт дослідження. Об’єктом дослідження є процес швидкості роботи цибулевої маршрутизації.

Предмет дослідження. Предметом дослідження є швидкість підключення до системи Tor і способи її вимірювання та підвищення.

Методи дослідження. Методами дослідження є аналіз інформаційних джерел, новітніх статей за темою дослідження, стандартів мережних протоколів, та здійснення експерименту із використанням власного програмного забезпечення.

Новизна одержаних результатів. Розроблено програмне рішення для визначення швидкості підключення до мережі. Дане рішення визначає статистичні дані про швидкість та умови підключення системою Tor.

Практичне значення одержаних результатів. Практичною цінністю роботи є використання програмного рішення для забезпечення подальших досліджень факторів, що впливають на швидкість користування системою Tor та збільшення цієї швидкості. Розроблене рішення може бути використано розробниками та дослідниками криптографічних мереж із забезпечення швидкісного підключення.

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа имеет объем 60 страниц, содержит 10 рисунков, а также 45 библиографических источников.

Проблему анонимности пользователя решают криптографические сети. которые пытаются предотвратить анализа трафика, смешивая сообщения, проходящие через сеть, таким образом, чтобы внешний наблюдатель не смог отследить источник и пункт назначения сообщения. Но, несмотря на то, что такие сети работают не один десяток лет, их скорость все еще остается невысокой. В данной работе поставлена цель проанализировать скорость популярной сети Tor и определить факторы, на нее влияющие.

Актуальность работы. Тема актуальна и на сегодняшний день, ведь анонимная сеть насчитывает большое количество пользователей, требующих стабильной и быстрой работы.

Цель и задачи исследования. Целью и задачей работы является исследование принципов луковой маршрутизации, механизмов системы Tor, измерения скорости передачи данных и способов ускорения работы луковой маршрутизации.

Объект исследования. Объектом исследования является система Tor.

Предмет исследования. Предметом исследования является скорость подключения к системе Tor и способы ее измерения и повышения.

Методы исследования. Методами исследования является анализ информационных источников, новейших статей по теме исследования, стандартов сетевых протоколов, и осуществления эксперимента с использованием собственного программного обеспечения.

Новизна исследования. Разработано программное решение для определения скорости подключения к сети. Данное решение определяет статистические данные о скорости и условиях подключения системой Tor.

Практическое значение полученных результатов. Практической ценностью работы является использование программного решения для

обеспечения дальнейших исследований факторов, влияющих на скорость пользования системой Tor и увеличение этой скорости. Разработанное решение может быть использовано разработчиками и исследователями криптографических сетей по обеспечению скоростного подключения.

SUMMARY

Thesis has a volume of 60 pages, contains 10 figures and 45 bibliographic sources

The problem of user anonymity is solved by cryptographic networks. which try to prevent traffic analysis by mixing messages passing through the network so that an outside observer cannot track the source and destination of the message. But, despite the fact that such networks have been operating for more than a decade, their speed still remains low. This paper aims to analyze the speed of the most popular network Tor and determine the factors that affect it.

Relevance of work. The topic is relevant today, because the anonymous network has a large number of users who require stable and fast work.

The purpose and objectives of the study. The purpose and task of the work is to study the principles of onion routing, the mechanisms of the Tor system, measuring the data rate and ways to speed up the work of onion routing.

Object of study. The object of study is the Tor system.

Subject of study. The subject of the study is the speed of connection to the Tor system and ways to measure and increase it.

Research methods. The research methods are the analysis of information sources, the latest articles on the research topic, the standards of network protocols, and the implementation of the experiment using its own software.

Novelty of the obtained results. A software solution for determining the speed of network connection has been developed. This solution determines the statistics on the speed and connection conditions of the Tor system.

The practical significance of the obtained results. The practical value of the work is the use of a software solution to provide further research into the factors that affect the speed of use of the Tor system and increase this speed. The developed solution can be used by developers and researchers of cryptographic networks to provide high-speed connection.

ЗМІСТ

ВСТУП

Протягом усієї історії розвитку та еволіції цифрових мереж ми спостерігаємо чітку тенденцію - зробити користування ними, з одного боку, швидким та безпечним, а з іншого – задовольнити потребу користувачів у збереженні анонімності. Для забезпечення безпеки користувача, що бажає бути анонімним, було розроблено декілька систем та підмереж Інтернету, але їх швидкість роботи є значно меншою за глобальну мережу.

Швидкість з’єднання є найважливішим аспектом у користуванні глобальною мережею, оскільки саме від цього параметру залежить зручність користування різними сайтами та сервісами, можливість повною мірою використовувати сучасні веб-додатки, що є вимогливими до швидкості передачі даних.

Таким чином актуальність розвитку технологій визначення швидкості з’єднання з мережею TOR буде в подальшому впливати на її зручність та популярність.

Мета даної роботи: підвищення точності визначення швидкості підключення до зашифрованої мережі.

Відповідно до поставленої мети в роботі сформульовані наступні задачі:

- дослідити методи визначення швидкості з’єднання;
- розробити алгоритм розрахунку швидкості передачі даних за допомогою різниці в часі надходження та відправки файла на сервер.
- розробити імітаційну модель системи визначення швидкості передачі

Об’єкт дослідження: швидкість передачі даних

Предмет дослідження: передача великого обсягу даних, побудова «цибулевого тунелю», створення власного сервера в мережі TOR.

Особистий внесок студента є розробка алгоритму визначення швидкості передачі даних та впровадження даного алгоритму в програмному вигляді.

РОЗДІЛ 1

МЕРЕЖА TOR

1.1. Захист інформації

З часу винаходу надсилання повідомлень між людьми поштою виникає необхідність запобігати отриманню та прочитанню одних і тих самих повідомлень іншими. Коли вік телекомунікацій та Інтернету настав разом із фізичним надсиланням повідомлень, проблема, звичайно, зросла. Великі зусилля докладаються для боротьби з повідомленнями та розробки ефективних способів шифрування повідомлень, щоб лише ті, хто бере участь у спілкуванні, знали значення цих повідомлень. Зазвичай повідомлення, що передаються через Інтернет, шифруються, і всі знають лише відправника та одержувача, але в деяких випадках навіть цю інформацію просять зберігати приватною від інших.

Термін аналіз трафіку використовується для опису процесу, який намагається збирати та аналізувати повідомлення, що проходять через інформаційну мережу, з метою зробити висновки про трафік на основі зібраної інформації. Кожна дія в Інтернеті надсилає ряд повідомлень через безліч серверів, які можуть бути розташовані в будь-якій точці світу, що робить дію досить публічною. Це означає, що комусь порівняно легко відстежувати такий тип трафіку. Не важко прослуховувати дорожній рух. Оскільки дані передаються через Інтернет, вони проходять через декілька місць, і тому, хто намагається підслуховувати ці повідомлення, потрібно лише знаходитись в одному з цих небагатьох місць на шляху зв'язку, куди передаються повідомлення, щоб контролювати трафік.

IP-пакет, що передає дані через Інтернет, складається з двох частин: по-перше, передані дані, які називаються корисним навантаженням, і по-друге, заголовок, що містить метадані корисного навантаження. Корисне навантаження зазвичай зашифровується перед відправкою, але частина заголовка не зашифрована і багато говорить про повідомлення, такі як відправник, одержувач,

час відправлення та розмір пакета. Це створює проблему інформаційної безпеки для приватних осіб та організацій, а також усіх, хто втручається в них взагалі.

Згідно з описаним вище, все більше і більше користувачів Інтернету не хочуть, щоб незнайомці бачили, що вони роблять в Інтернеті, і тому існує попит на програми, які можуть запропонувати повну анонімність. Користувачі та причини використання таких додатків дуже різноманітні: звичайні люди хочуть захистити своє приватне життя від маркетологів та безвідповідальних компаній, журналісти та їхня аудиторія хочуть гарантувати свободу слова без політичної цензури, правоохоронні органи хочуть уважно вивчити кіберзлочинність та військову діяльність. хочуть захистити себе від шпигунства ворога під час операцій.

Історія перенаправлення цибулі сягає 1995 року, коли Управління морських досліджень (ONR) у Сполучених Штатах розпочало фінансування проекту з вилучення особи з диверсії. Іншими словами, їх метою було створити спосіб анонімного зв’язку в Інтернеті [1]. Спочатку троє людей, Девід Голдшлад, Майкл Рід та Пол Сіверсон, почали розробляти технологію, пізніше відому як цибульна маршрутизація. Першим результатом проекту став перший прототип у 1996 році, після чого було розроблено ще два покоління цієї технології.

До того, як Голдшлад, Рід та Сіверсон почали розробляти перші прототипи, вони мали уявлення про те, якою буде ця технологія. По-перше, код повинен бути загальнодоступним. Це дозволяє користувачам, які використовують цю систему, бачити код, щоб довіряти системі. Другим основним принципом управління цибулею є можливість вільного пересування. Це означає, що комп'ютер, який використовується для доступу до системи, не обов'язково повинен бути сервером.

Ця політика має деякі переваги безпеки, які будуть проаналізовані пізніше.

Цибульна маршрутизація отримала свою назву завдяки структурі даних, в якій інформація розподіляється по мережі [2]. Дуга створюється із шарів, які криптографічно зашифровані відправником повідомлення. Кожен

маршрутизатор на комунікаційному шляху очищає ці рівні по черзі, коли повідомлення проходить через мережу. Вхідні повідомлення йдуть тим же шляхом, але маршрутизатори додають шари до цибулі, а потім очищаються стороною, що приймає.

Як вже згадувалося раніше, цибульна маршрутизація покладається на створення криптографічного шляху між двома комп'ютерами через кілька маршрутизаторів. Цей шлях називається цибулевим ланцюгом [1]. Маршрутизатори, що формують маршрут, випадковим чином вибираються з усіх серверів, зареєстрованих у цибульній мережі, і схема генерується таким чином, що кожен вузол знає лише наступний і попередній маршрутизатори на шляху зв'язку. Ініціатор шляху вирішує, які маршрутизатори включати в ланцюжок.

Щоб усі маршрутизатори на комунікаційному шляху не відкривали повідомлення, що проходять через ланцюг, ініціатор повинен окремо розподілити унікальні симетричні ключі шифрування до кожного відповідного маршрутизатора. Причиною використання симетричних ключів є те, що симетричне шифрування використовує один і той же ключ для шифрування та дешифрування повідомлень, тому повідомлення можна надсилати в обох напрямках одним і тим же ключем. Симетричне шифрування також обчислювально набагато простіше, ніж шифрування відкритим ключем.

Ключі симетричного шифрування поширюються за допомогою шифрування із відкритим ключем. Криптографія з відкритим ключем використовує два ключі: приватний ключ (невідомий іншим) і відкритий ключ (відомий всім). Повідомлення, надіслані одержувачу, шифруються за допомогою чіткого повідомлення, яку, у свою чергу, не можна використовувати для розшифровки повідомлень. Одержувач може відкрити повідомлення своїм приватним ключем. Оскільки кожен маршрутизатор має власний відкритий ключ, симетричні ключі, що використовуються для створення дуги, можна надійно розподілити за допомогою відкритих ключів без ризику виявлення іншими маршрутизаторами.

Причиною використання симетричного шифрування для створення дійсної дуги є те, що симетричне шифрування вимагає набагато меншої обчислювальної потужності, ніж шифрування відкритим ключем. У мережі багато повідомлень, що рухаються у всіх напрямках, і кожне повідомлення має оброблятися за допомогою алгоритму шифрування. Використання симетричного шифрування замість шифрування із відкритим ключем значно полегшує обчислювальні витрати на мережу.

Алгоритм створення цибулевого ланцюжка детально описаний у наступному розділі. Далі ми детальніше розглянемо розвиток технології цибульної маршрутизації. Як уже згадувалося раніше, маршрути цибулі розроблялися поетапно протягом декількох поколінь. Зазначені поки функції не стосуються всіх трьох поколінь, але вони включені в технологію принаймні одного. Відмінності між поколіннями, що розвиваються, будуть головною темою наступного параграфа.

1.2. Виникнення мережі

Коли в американській морській дослідницькій лабораторії (NRL) було розпочато проектування маршруту цибулі, Голдшлад, Рід та Сіверсон мали лише декілька основних принципів та деякі припущення щодо ситуацій, в яких технологія буде використана для керівництва ними у їх роботі. Але коли було випущено перше покоління, вони отримали свій перший зразок того, наскільки функціональним був насправді їх розвиток. Додані та вдосконалені функції в наступних поколіннях на основі спостережень. Отже, покоління представляють різні етапи розвитку маршруту цибулі, і, досліджуючи ці покоління, можна зрозуміти, чому саме цибулеву технологію застосовували, як сьогодні.

1.2.1 Перше покоління (1996-2004)

Основна ідея цибульної маршрутизації полягає в тому, щоб зробити можливим встановлення анонімного з'єднання в цибульній мережі із серверами поза цибулевою мережею. Також було вирішено встановити зв'язок між двома

маршрутизаторами, підключеними до цибульної мережі. В обох цих випадках довжина ланцюга, а саме кількість маршрутизаторів у ланцюжку, включаючи відправника та одержувача, має велике значення з точки зору захисту інформації.

Щоб один із вузлів ланцюжка порушив анонімність, він повинен підключити анонімні повідомлення до відправника та одержувача. В основному, теоретичний зловмисник повинен отримати IP-адресу відправника та одержувача. Зазвичай це можна зробити, переглядаючи заголовки пакетів, але цибулеве перенаправлення шифрує ці заголовки. Основна атака на маршрутизацію цибулі полягає в тому, щоб встановити маршрутизатор як цибульний вузол і спробувати контролювати повідомлення, що проходять через цей маршрутизатор.

Якщо довжина ланцюжка становить три вузли (відправник, цибульний маршрутизатор та одержувач), а середній вузол - вузол зловмисника, який хоче знайти IP-адреси двох кінців ланцюжка, він негайно дізнається ці адреси. При використанні чотирьох вузлів (відправник, два маршрутизатори, приймач) зловмисникові потрібно буде лише розшифрувати один з вузлів, щоб порушити анонімність ланцюжка. Розробники вважали, що це досить небезпечно, тому вони вирішили встановити мінімальну довжину (довжина ланцюжка за замовчуванням), рівну п'яти вузлам.

Так звані сервери знайомств використовуються, якщо дві машини, підключені до цибульної мережі, хочуть надсилати повідомлення один одному. Машини не можуть створити шлях підключення на основі IP-адрес один одного, тому сервер зустрічей використовується як посередник для розповсюдження повідомлень. Оскільки обидві сторони повинні створити власні ланцюжки цибулі між серверами зустрічей, довжина шляху, включаючи всі маршрутизатори, становитиме дев'ять вузлів. Ця особливість залишалася для наступних поколінь.

Найбільша різниця між першим поколінням і наступними двома полягає в тому, що у версії першого покоління клієнтське програмне забезпечення та

цибульний маршрутизатор повністю інтегровані. Його також можна додати як частину цибульної ланцюга, оскільки комп'ютер підключений до мережі для надсилання повідомлень. Ця функціональність суперечила одному з основних принципів цибульної маршрутизації і була змінена після випуску другого покоління, що дозволило запустити клієнт і цибульний маршрутизатор окремо.

1.2.2 Друге покоління (2004-2006)

Для підвищення інформаційної безпеки до другого покоління додані різні методи. Одним із методів є надсилання повідомлень із випадковими даними, які називаються відступами, разом із звичайним трафіком, щоб зловмиснику було важче проаналізувати трафік. Другий метод полягає в обмеженні смуги пропускання до постійної швидкості, що виключає виявлення змін у потоках транспорту. Вважалося, що доповнення та обмеження пропускної здатності ускладнюють аналіз трафіку, але зазначено, що ці методи обчислювально дуже дорогі порівняно із захистом, який вони забезпечують. В третьому поколінні були скасовані обмеження пропускної здатності та обмеження.

Деякі криптомережі, які пропонують анонімність, покладаються на змішування компонентів, а не на непередбачуваний спосіб. Ці мережі також відомі як мережі MIX (розроблені Девідом Чаумом у 1981 р.) [2]. Ціль змішування - ускладнює зв'язок зашифрованих вихідних повідомлень із вхідними повідомленнями.

Технологія MIX експериментально була включена в цибульну маршрутизацію другого покоління. Метою цього експерименту було вивчити переваги скремблювання для безпеки.

Передбачалося, що прийняття вже широко використовуваного методу зробить маршрутизування цибулі більш прийнятним для широкої аудиторії.

У третьому поколінні перемішування остаточно зупинили з тих же причин, що і наповнення та обмеження місткості; це недостатньо покращило інформаційну безпеку та суттєво збільшило вартість обчислень. Виконання фактичного змішування не вимагає великих обчислювальних витрат, але

додавання суміші створює ризик того, що хтось прослуховує повідомлення, і дотепер не знайдено дешевого способу запобігти цьому.

Окрім перетасовки, ще однією важливою зміною було створення ланцюгів довжиною більше п’яти вузлів. У цій версії покоління ланцюги варіювались у довжину і могли мати до одинадцяти вузлів у ланцюгу. Шляхом тунелювання ланцюгів або об’єднання кількох ланцюгів довжина шляху може бути додатково збільшена. Як і у схемах змішування та наповнення, ця зміна була визнана непотрібною, і кількість вузлів нижче було відновлено до п'яти.

1.2.3 Третє покоління (2006 -)

По-третє покоління - остання версія перенаправлення цибулі. Розвитку маршрутна цибуля все ще триває, тому майбутні можливості цибульної маршрутизації можуть відрізнятися від представлених тут. Однак ми розглянемо особливості третього покоління, як воно було опубліковане в 2006 році. Відмінності між цим поколінням та попереднім поколінням менші,

ніж відмінності між першим та другим поколіннями.

Найбільшим виправленням порівняно з попередніми версіями є протокол, де ключі шифрування розподіляються по вузлах при першому створенні каналу. Перша версія використовувала структуру даних onion-data для створення шляху [1]. Ця версія використовує протокол Діффі-Хеллмана для поступового розподілу ключів між вузлами. Це безпосередньо забезпечує конфіденційність; це означає, що навіть якби зловмисник зміг розшифрувати та отримати один із ключів шифрування, вони не змогли б використовувати його для відкриття інших рівнів цибулі.

Ще однією зміною цієї версії стало додавання серверів каталогів. До цієї зміни інформація про мережеву адресу надавалась із мережі зовні. У міру збільшення розміру мережі такий спосіб надання інформації стає неможливим, а сервери каталогів пропонують гнучкий спосіб вирішення цієї проблеми.

Це також підвищує безпеку, оскільки розподілену інформацію неможливо вивчити. У наступному розділі детальніше розглядається фактичне застосування технології цибулі третього покоління.

1.3. TOR

Перша версія маршруту цибулі була опублікована в 1996 році [1]. Коли Управління морських досліджень фінансувало проект, вони також підтримували веб-сайт, на якому розміщувалася інформація про хід проекту. Цей сайт містив як наукову, так і технічну інформацію про різні покоління.

Статті про маршрутизацію цибулі. Оскільки ONR припинив фінансування в 2004 році, веб-сайт також було припинено, а сайт закрито. Однак копія сайту доступна за адресою www.onionrouter.net [3].

Коли ONR припинив фінансування, Фонд Electronic Frontier, або незабаром EFF, розпочав фінансування проекту. Полу Сіверсону та команді розробників цибулевого корму вдалося запустити три покоління маршрутів цибулі за фінансування ONR. Коли спонсор змінився, було зроблено перехід на відкритий код, а разом з ним нову домашню сторінку було переміщено на www.torproject.org [4].

Разом з новим сайтом також було випущено додаток під назвою Tor. Мета Tor - створити мережу, яка захищає користувача від автентифікації за допомогою технології цибульної маршрутизації. Іншими словами, Tor - це практична реалізація цибулевої експедиції. З цього моменту, коли ми обговорюємо перенаправлення цибулі, ми говоримо про Tor, про реалізацію перенаправлення цибулі третього покоління, і навпаки.

1.3.1 Tor браузер

Tor написаний мовою програмування C, а його вихідний код - у відкритому сховищі за адресою gitweb.torproject.org/tor.git. Для використання програми будь-який користувач може завантажити пакет із назвою Tor Browser Bundle або TBB з веб-сайту проекту [4]. TBB включає програму, яка дозволяє користувачеві

підключатися до мережі Tor як клієнт, та спеціальний браузер Firefox, який можна використовувати для відвідування веб-сайтів.

TBB містить клієнта та необхідне програмне забезпечення для роботи реле. TBB попередньо налаштовано на роботу в якості клієнта, і користувачеві не потрібно вводити додаткові налаштування для використання програмного забезпечення [4]. Однак є кілька дій, які користувач може вжити для вдосконалення.

Конфіденційність, наприклад, увімкнення розширення під назвою NoScript, яке відключає JavaScript та використання HTTPS лише для підключення.

Щоб додати нове реле Tor до мережі, користувач повинен відредагувати файл конфігурації з інформацією про порти та політикою виходу.

Клієнти використовують програмне забезпечення Tor для створення з’єднань через реле. Всі комп'ютери, підключені один до одного за допомогою TBB, утворюють мережу, яка називається мережею Tor. Незалежно від клієнтів та реле, для ефективної роботи мережі Tor потрібні інші типи компонентів. Давайте подивимось, які ще типи мережевих компонентів Tor.

1.3.2 Tor мережа

У найпростішій формі реалізація цибульної мережі маршрутизації Він складається з клієнтів, які використовують мережу для надсилання повідомлень, та маршрутизаторів або ретрансляторів, які приймають цибулю, очищають її та пересилають до наступного вузла в ланцюжку.

Окрім клієнтів та повторювачів, мережа Tor має, наприклад, сервери імен, які контролюють сервери, зареєстровані в мережі, та приховані служби, які можна використовувати, наприклад, для веб-трансляції.

Приховані служби можуть підключатися лише через певні сервери Tor.

Клієнти та маршрутизатори є основною частиною мережі Tor [5]. Клієнти також можуть виступати в ролі маршрутизатора в мережі, але більшість клієнтів використовують мережу лише для підключення до інших клієнтів, прихованих служб та зовнішніх служб через інші маршрутизатори. У березні 2016 року

кількість клієнтів, що користуються мережею, становила приблизно 2 мільйони, а в 2016 році кількість повторювачів мережі становила 7000.

Щоб Tor не підслуховував, до мережі були додані захисні ворота разом із традиційними цибулевими маршрутизаторами. Призначення захисних пристроїв - запобігти атакам, коли зловмисна сторона намагається розмістити маршрутизатор як перший вузол у ланцюжку. На вихідних реалізація першого вузла була вибрана випадковим чином серед усіх маршрутизаторів, підключених до мережі; це означало, що поки зв’язок встановлювався неодноразово протягом достатнього часу, в певний момент зловмисний маршрутизатор буде обраний першим вузлом.

Починаючи з версії 2006 року, із невеликої кількості надійних маршрутизаторів замість одного вузла введення вибираються три вузли безпеки. Ці три вузли безпеки будуть використовуватися для всіх каналів, створених замовником, протягом 30 - 60 днів перед тим, як відмовитись і перейти до вибору нових вузлів входу [7]. Це значно збільшує здатність мережі протистояти атакам підслуховування.

Іншим заходом підвищення безпеки є додавання так званих гіперпосилань [1]. Для поліпшення безпеки деякі маршрутизатори не публічно оголошували свої адреси. Мости виникають через те, що загальнодоступні IP-адреси Tor заблоковані деякими брандмауерами. Людина, яка не має доступу до мережі Tor через такі брандмауери, може використовувати мости, щоб обійти ці фільтри.

Про спецслужби також згадувалося раніше. Таким чином вони не підвищують інформаційну безпеку, але розробники хотіли включити спецслужби. Ці послуги можна використовувати, наприклад, для надсилання миттєвих повідомлень між клієнтами. Коли секретна служба хоче опублікувати свою адресу, вона створює кілька лампочок-каналів, а ім'я повідомляє адресу останнього маршрутизатора службового каналу. Клієнт може підключитися до послуги через вузли зустрічі.

1.3.3 Створення ланцюжка

Давайте детальніше розглянемо, як формується ланцюжок. Коли користувач Tor хоче підключитися до сервера, який не є Tor, клієнтське програмне забезпечення спочатку завантажує список усіх зареєстрованих маршрутизаторів Tor [6]. З цих маршрутизаторів спочатку вибирається вузол введення.

Вузол введення є першим маршрутизатором у цибульному ланцюжку і вибирається зі списку маршрутизаторів, про які служба імен швидко повідомляє, і чесний. Якщо вибраний вхідний вузол не існує, вибирається новий вихідний вузол.

Після вибору вхідного вузла створюється інша частина схеми. Вузли, перераховані з найбільшою пропускною здатністю та часом безвідмовної роботи, частіше додаються до каналу. Вибираються лише вузли, визначені як стабільні, щоб зробити ланцюг максимально стабільним. На рисунку 1.1 показаний псевдо-алгоритм вибору вихідного маршрутизатора та середнього маршрутизатора в ланцюжку [6]. Алгоритм відрізняється від поточної практики тим, що маршрутизатор не може оголосити пропускну здатність сам, але пропускна здатність вимірюється всередині мережі [8].

Алгоритм: Вибір вхідного маршрутизатор

Вхід: Список відомих Tor маршрутизаторів, router_list

Вихід: псевдовипадково обраний маршрутизатор, зважений по відношенню до маршрутизатора з максимальною пропускною спроможністю

B ← 0, T ← 0, C ← 0, i ← 0, router_bw ← 0, bw_list ← Ø

foreach router r ϵ router_list **do**
  router_bw ← get_router_adv_bw(r)
  B ← B + router_bw
  bw_list ← bw_list ∪ router_bw
end

C ← random_int(1,B)

while T < C **do**
  T ← T + bw_list_i
  i ← i + 1
end

return $router\_list_i$

Рис. 1.1 Псевдокод: Алгоритм вибору маршрутизатора без вводу

- B = комбінована пропускна здатність усіх маршрутизаторів,
- T = сума смуг пропускання в підциклі,
- C = випадково вибране число з B,
- i = кількість обробленої в даний момент смуги пропускання в підпрограмі
- loop, router_bw = оголошена пропускна здатність, яка обробляється у верхньому циклі,
- bw_list = список, що містить всю пропускну здатність маршрутизатора.

Після випадкового вибору трьох маршрутизаторів створюється ланцюжок, один за одним, генеруючи ключі шифрування.

Будь-який маршрутизатор, який використовує протокол Діффі-Хеллмана. Нові пари ключів поступово відображаються на кожному вузлі після завантаження ключів з попереднім вузлом у шляху. Це забезпечує так звану пряму конфіденційність; це означає, що жоден вузол не може простежити ключі, які інші вузли використовують для відкриття повідомлень. Після цієї процедури кожен маршрутизатор має власний ключ шифрування для створення та очищення цибулі, а також інформацію про те, кому надсилати повідомлення. В формі 1.2 показано посилання на службу імен та цибульний шлях після його появи.

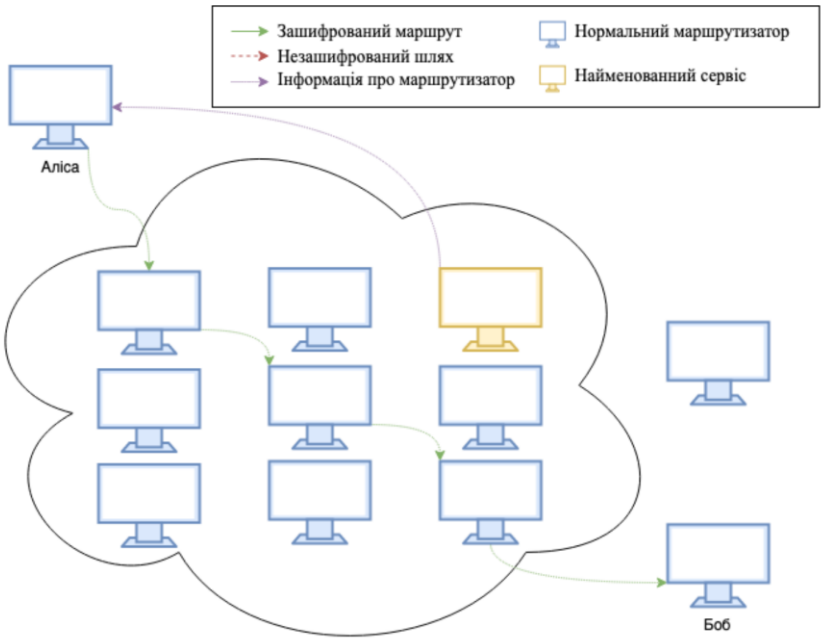


Рис. 1.2 Створення цибульного шляху та взаємодія зі службою імен

Клієнт Аліса завантажує список цибульних маршрутизаторів із служби імен і створює шлях. Вихідний вузол надсилає Бобу незашифроване повідомлення.

В результаті створення окремих маршрутів з кожним маршрутизатором окремий вузол не може знати адреси всіх інших вузлів у ланцюжку. Перший маршрутизатор бачить, що клієнт підключений до мережі, але не кінцевий пункт призначення повідомлень, які користувач надсилає через Tor. Вузол входу не може знати, що користувач робить з мережею. Він отримує повідомлення і відправляє їх вперед ланцюжком. По-третє, кінцевий вузол може бачити, куди були надіслані повідомлення, але не може бачити, хто їх надіслав, і які дані містяться в повідомленні. Третій вузол називається вузлом виходу. Вихідний вузол надсилає повідомлення кінцевому одержувачу.

Останній вузол у ланцюжку, який отримує незашифроване повідомлення від вихідного вузла, може не знати, що він є частиною цибульного ланцюга, оскільки повідомлення залишає вихідний вузол незашифрованим. Тільки порівнявши інформацію про відправника в заголовку зі списком відомих маршрутизаторів Tor (список зберігається на torstatus.blutmagie.de), одержувач може визначити, що повідомлення надходять з цибульної мережі. Багато Інтернет-служб можуть блокувати трафік через адреси Tor.

Висновки до частини 1

У цій частині роботи розглядаються походження та ідея цибулевого управління, технологічний огляд цибулевого управління та еволюція технологій протягом трьох поколінь.

Наведено практичну реалізацію використання перенаправлення цибулі на ім'я Tor. Проаналізовано його властивості, алгоритм створення цибулевого ланцюга, проходження повідомлення в ланцюгу цибулі та врахування значення клітин у системі.

Таким чином, програмний додаток цибульного перенаправлення під назвою Tor був створений для забезпечення повної анонімності користувача та його дій в Інтернеті. Використовуючи програмне забезпечення Tor, користувач забезпечує повну анонімність дій в Інтернеті завдяки перенаправленню цибулі.

РОЗДІЛ 2

ВИМІРЮВАННЯ ШВИДКОСТІ ПІДКЛЮЧЕННЯ

Найактуальнішою проблемою системи TOR є повільна швидкість з’єднання. Саме аналіз цих даних і є предметом дослідження Крім того, потрібно врахувати деякі фактори, які впливають на швидкість передачі даних і на результат її тестування. Методів є декілька, тож почнемо від простіших.

Для початку розберемося з тим, що може вплинути на якість вимірювання швидкості інтернет з'єднання. А таких факторів чимало:

- швидкість сервера, з яким виконується зв’язок, перевіряючи швидкість Інтернету;
- швидкість і налаштування роутера, якщо комп'ютер підключений до локальної мережі через нього;
- працюючі програми на комп'ютері в момент перевірки;
- антивіруси і брандмауери, що працюють у фоновому режимі;
- налаштування комп'ютера і операційної системи.

У більшості випадків найважливішим в цьому списку буде перший пункт - швидкість сервера. Давайте переглянемо декілька прикладів.

Якщо Ви підключите свій комп'ютер до локальної мережі за допомогою кабелю (витої пари), то швидкість з'єднання з іншим комп'ютером в цій же мережі (у Вашому місті) буде дуже велика, наприклад, 70 Мбіт/с.

Тепер заміряємо швидкість між своїм комп'ютером і яким-небудь сервером в іншому кінці країни. Можливо, отримаємо близько 20 Мбіт/с.

Уявімо, що до цього сервера підключилися одразу всі 800 відвідувачів, і почали завантажувати різні файли. Швидкість зв'язку з завантаженим сервером зменшиться і для Вас буде, наприклад, 3 Мбіт/с.

А тепер спробуємо завантажити файл з сервера в іншій країні, наприклад, в Австралії, отримаємо менше 1 Мбіт/с!

Тепер зрозуміло, що виміряна швидкість залежить від того, який сервер обрано для перевірки: чи впливає його розташування, його власна максимальна швидкість і його завантаженість. Тобто, потрібно зрозуміти, що швидкість у всіх серверів (що містять сайти і файли) різна і залежить від можливостей цих серверів.

Те саме можна сказати і до інших пунктів (наприклад, швидкість з'єднання буде відрізнятися при з'єднанні з мережею безпосередньо або через роутер, а також буде залежати від характеристик цього роутера).

2.1. Підвищення точності перевірки

Ці пункти бажано виконати, якщо потрібно отримати максимально точний результат перевірки швидкості інтернет з'єднання.

- підключити мережевий кабель безпосередньо до комп'ютера (в роз'єм мережевого адаптера);
- закрити всі програми, крім браузера;
- зупинити програми, які виконують завантаження в фоні (торрент-клієнти, менеджери завантажень і т.д.), крім тих, які обрані для тестування швидкості інтернету;
- тимчасово вимкнути антивірус, тому що в деяких випадках він може впливати на показання онлайн-тестів;
- переконатися, що мережа не завантажена («Використання мережі» має бути менше 1%). Якщо мережа активно використовується, можливо, відбувається оновлення будь-якої програми або Windows. В цьому випадку варто дочекатися закінчення завантаження або перезавантажити комп'ютер.

Крім того, кожний вимір потрібно провести кілька разів, щоб підвищити точність тестування.

2.2. Методи вимірювання швидкості підключення

Є кілька способів виміряти швидкість інтернет з'єднання. Розглянемо їх у порядку зростання складності використання. Це – використання онлайн-сервісів, передача файлів та використання торент-мережі для вимірювання.

Останній спосіб не передбачає підключення до мережі TOR, тому для даної роботи він не є актуальним. Розглянемо перші два способи детальніше.

2.2.1. Вимірювання швидкості за допомогою онлайн сервісів

Це той самий простий спосіб вимірювання «натиснути одну кнопочку і все швидко дізнатися». Точність відносна, але простота привертає. Зверніть увагу, що онлайн тести мають різну точність! Сьогодні розглянемо тільки найпопулярніші сервери.

Почнемо з самого популярного і найточнішого онлайн тесту для перевірки швидкості Інтернету (до того ж, безкоштовного) – Speedtest net. На ньому і зупинимось детальніше та в подальшому радимо вам використовувати саме його. Ось коротка інструкція:

1. Зайти на сайт онлайн-сервісу за посиланням: http://www.speedtest.net/ за допомогою цибулевого підключення

2. Використати кнопку «Почати перевірку» для запуску процесу вимірювання швидкості підключення до серверів сервісу через цибулеву мережу;

3. Після закінчення тестування проаналізувати три результати, що показує сервіс.

Перше число позначено словом «Ping», позначає час передачі мережевих пакетів. Чим менше це число, тим краще якість з'єднання (бажано менше 100 мс). По суті, даний показник означає власне швидкість передачі мережевих пакетів, а не пропускну здатність каналу зв’язку.

Друге число - швидкість отримання даних. Саме цей показник є найважливішою метрикою, оскільки воно показує швидкість прийому даних з сервера, а це – близько 70% користувацького трафіку.

Третє число - швидкість передачі даних на сервер. За результатами перевірок, воно може істотно відрізнятись від швидкості отримання даних, але корисне воно буде у 3% використання трафіку, адже воно показує швидкість відправки пакетів на сервер, а такий процес зазвичай є рідшим та менш громіздким за обсягом даних.

Якщо Ви хочете виміряти швидкість інтернет з'єднання з будь-яким конкретним містом, то виберіть його на карті (доступні всі великі міста планети) і знову натисніть кнопку «Почати перевірку».

2.2.2. Вимірювання швидкості за допомогою ручного завантаження файлу

Більш точним тестом, який показує реальну швидкість завантаження файлів, буде наступний варіант:

1. Встановити менеджер завантажень (ми радимо користуватися програмою Download Master);

2. Додати кілька завантажень з різних швидких файлових серверів;

3. Виставити в налаштуваннях максимальну кількість потоків (секцій для завантаження);

4. Стежити за максимальною швидкістю завантаження файлів.

Саме цей метод покладено у основу розробленого програмного продукту у даній роботі. Суть в тому, що за допомогою програмного рішення перевіряється довготривала швидкість завантаження великих об’ємів інформації з сервера. Така перевірка має ряд переваг над онлайн-рішеннями. По-перше, вона здатна тестувати швидкість з’єднання у великий проміжок часу, отже дає більш повну інформацію про стан підключення. По-друге, великий об’єм файлів, що передаються, додатково створює тестування на навантаженість мережі, що іще більше показує мережеву картину в цілому.

З використанням даного методу, можна заміряти швидкість цибулевого підключення протягом тривалого проміжку часу та на великий об’єм, що дозволяє вести досить повну, точну та глобальну статистику стану підключення.

2.3. Безпека користування мережею TOR

Метою цибулевої маршрутизації є запобігання локалізації та ідентифікації користувача. Це означає, що атаки на систему намагаються підключитися, наприклад, до журналів відвідування веб-сайту або відправки повідомлень назад користувачеві. У цьому розділі ми розглянемо, які атаки були розроблені проти Tor.

Яким чином потенційний зловмисник може спробувати деанонімізувати користувачів і як мережа призначена для захисту від подібних атак? Деякі атаки можуть мати на меті зниження якості обслуговування, наприклад, за допомогою атак типу «відмова в обслуговуванні», але в данній тезі ми виключаємо подібні атаки і концентруємося на деанонімізація атак.

Цибулева маршрутизація захищає від ідентифікації за допомогою заплутування. На практиці цибулева маршрутизація може забезпечити анонімність, тільки якщо її користувацька база досить велика. Чим більше користувачів в мережі, тим більше цілей для об'єднання певної активності. Це означає, що один користувач не може створити власну цибулеву мережу і створити собі анонімний захист. Сила Tor в тому, що зловмисники не можуть простежити шлях повідомлень, а також відстежити повідомлення, які виходять з мережі, тому, хто їх відправляє.

Це як мінімум концепція системи. Існує безліч різних атак на криптографічні системи. Більшість атак на Tor засновані на спробі захопити один або два маршрутизатора в мережі. Зазвичай, принаймні, один маршрутизатор вузла входу в ланцюзі і, можливо, також вихідний вузол в тому же ланцюзі. За допомогою цієї настройки можна виконати атаку аналізу трафіку, яка намагається зв'язати вхідні і вихідні повідомлення. Пасивні атаки відстежують

потік повідомлень, не втручаючись в нього, а активні атаки якимось чином модифікують трафік, щоб спростити аналіз. У цьому розділі ми розглянемо, які інші типи атак призначені для системи.

Виділяють п'ять категорій атак: імовірнісні моделі, атаки вибору маршрутизатора входу і виходу, атаки на рівні AS і на глобальному рівні, атаки на основі аналізу трафіку і часу, а також вразливості протоколів. У своїй статті Сало перераховує ці категорії і описує раніше опубліковані атаки, які відносяться до цих категорій. Давайте коротко розглянемо їх.

Імовірнісні моделі використовують математичні імовірнісні моделі для аналізу мережі. Ці моделі спочатку були призначені для MIX-мереж і використовують Байєсовські імовірнісні моделі, щоб виміряти, наскільки безпечна мережа. Такі ж моделі були розроблені для Tor. Моделі розглядають систему як чорний ящик, намагаючись створити математичні моделі, засновані на двох припущеннях:

• По-перше, що тільки один користувач пов'язаний з одним виходом.

• По-друге, що вихід може бути пов'язаний з користувачем у тому випадку, якщо можна спостерігати обидва виходи.

Ці моделі насправді не є атаками, а скоріше пропонують спосіб оцінити, наскільки можливо для зловмисника деанонімізувати користувача.

Атаки на вибір вхідного і вихідного маршрутизатора націлені на підвищення ймовірності вибору маршрутизатора атакуючого як вузол входу і/або виходу. Сало згадує про два типи атака з цієї категорії:

• По-перше, компрометація анонімності з використанням пакетного обертання, намагається створити петлі між маршрутизаторами, щоб вони могли відмовитися обслуговувати інших клієнтів, на практиці вбиваючи маршрутизатор. Мета полягає в тому, щоб збільшити ймовірність того, що зловмисні маршрутизатори будуть обрані в якості частини певного каналу.

• По-друге, низько ресурсною маршрутизацією проти Tor, які намагаються повідомити неправдиву інформацію про пропускну здатність і часу роботи

маршрутизатора службі імен, що дає маршрутизаторам більший пріоритет при виборі вузлів входу.

Атаки, засновані на аналізі трафіку і часу, охоплюють найбільшу кількість атак в категорії. Атаки такого роду намагаються послабити анонімність, спостерігаючи закономірності в трафіку, наприклад, під час мережевого потоку. За допомогою цих шаблонів зловмисник може зіставляти вхідний і вихідний мережевий трафік. Пасивні атаки виявляють, що пакети проходять через систему, а активні атаки намагаються, наприклад, помітити пакети водяними знаками, щоб їх було легше аналізувати.

У цій категорії сім дослідницьких робіт:

- Low-Cost Traffic Analysis of Tor представляє спосіб відстеження вузлів, які використовуються в ланцюзі, а також спосіб зв'язати незв'язані потоки з їх ініціатором на основі різних затримок в потоках.
- A cell counter based attack against Tor представляє техніку аналізу трафіку шляхом додавання водяних знаків для лічильника комірця.
- Browser-Based Attacks on Tor використовують атаку «людина-посередник» і модифікують HTTP-трафік для поліпшення аналізу.
- A Practical Congestion Attack on Tor Using Long Paths використовує атаку перевантаження з модифікацією HTTP потоку для вивчення шляхів каналів.
- How Much Anonymity does Network Latency Leak? Вимірює затримки в мережі для розпізнавання користувача.
- Passive Logging Attacks Against Anonymous Communications Systems спостерігає за поведінкою користувача і намагається передбачити ініціатора певного шляху.

AS і атаки глобального рівня припускають, що зловмисник може спостерігати або маніпулювати значною частиною трафіку, який входить і виходить з мережі. Цей вид зловмисника може використовувати, наприклад, аналіз трафіку і часу для розпізнавання потоків даних. Tor не призначений для

захисту користувача від AS (мається на увазі автономна система) рівня супротивника. Сало згадує дві статті про AS - усвідомлення в Tor Path Selection, які намагаються оцінити, наскільки ймовірний насправді зловмисник на рівні AS, і Large Scale Simulation of Tor, яке оцінює кілька атак аналізу трафіку в модельованій мережі.

Вразливості протоколу - остання категорія. Ці атаки намагаються знайти слабкі сторони в протоколах зв'язку. Ефективні атаки в протоколі аутентифікації Tor припускають вразливість в способі розподілу сеансових ключів, яка призведе до невідповідностей в сеансових ключах. Про ризик обслуговування, коли ви займаєтеся серфінгом намагається розпізнати мости в мережі Tor. Спочатку зловмисник намагається розпізнати можливих міст-кандидатів, а потім намагається підтвердити результати активною атакою на засмічення ланцюга.

2.4. Тестування ПЗ

Тестування є важливим кроком у життєвому циклі розробки програмного забезпечення. Тест можна проводити на різних рівнях.

Незалежно від використовуваних моделей, всі рівні тестування є важливими і методології.

Модульний тест. Цей рівень тестування покликаний перевірити роботу окремого елемента системи. Елемент - модуль системи визначається контекстом. Найбільш повний опис такого тестування наведено в IEEE 1008-87 "Стандарт програмного забезпечення". Unit Testing ", що визначає систематичну та інтегровану концепцію, задокументований підхід до модульного тестування.

Інтеграційне тестування. Цей рівень тестування - процес управління взаємодією між програмними компонентами / модулями. Класичні стратегії інтеграційного тестування - "зверху вниз" і "знизу вгору" - традиційні, ієрархічно структуровані системи та їх важко реалізувати, наприклад для тестування слабозв'язаних систем, вбудовані в сервісно-орієнтовані архітектури (SOA).

Тест системи. Тестові кришки системи - ціла система. Більшість функціональних збоїв визначається на рівні модульних та інтеграційних тестів. В порядку, тестування системи, часто зосереджуючись на нефункціональних вимогах - безпека, продуктивність, точність, надійність тощо. На цьому рівні інтерфейси до зовнішніх додатків, обладнання, робоче середовище тощо

Тестові мішені. Випробування проводять відповідно до зазначених умов. цілі (які можуть бути вказані явно або неявно) та різні рівні точності. Правильне визначення цілі дозволяє контролювати результати. тестування. Тестові кейси можуть бути розроблені для перевірки функціональності для оцінки вимог (відомих як функціональні тести) та функціональності вимоги. Однак існують тести, коли є кількісні параметри та результати, тести можуть лише опосередковано згадувати цілі задоволення тестування (наприклад, "зручність використання" - зручність, простота використання, не можуть бути чітко визначені кількісно в більшості випадків особливості).

Висновки до частини 2

Вимірювання швидкості підключення за допомогою завантаження великого об’єму даних із сервера впродовж тривалого часу – найоптимальніша та найбільш точна модель вимірювання цибулевого підключення. З’ясували, що дане підключення є досить безпечним і навіть атака часу та об’єму має дуже малу ймовірність у нашому випадку.

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1. Постановка завдання

За даними попередніх розділів, необхідно розробити програмне рішення, скероване на замір швидкості підключення до сервера через систему TOR. Для цього, необхідно виконати огляд існуючих програмних рішень, що імплементують дану задачу, визначити їх технології роботи, спільні та відмінні риси. Далі, необхідно обрати технологію розробки програмного забезпечення, визначити основні вимоги до ПЗ та проаналізувати результати, що отримані під час роботи готового програмного продукту.

В нашому випадку, окрім моделювання відправки та отримання файлу через систему TOR, існує необхідність у виведенні на екран результатів моделювання у графічному вигляді, тому для розробки програмного забезпечення було обрано мову програмування Python.

Python - інтерпретована мова програмування загального призначення високого рівня. Філософія дизайну Python підкреслює читабельність коду завдяки помітному використанню значних відступів. Його мовні конструкції, а також об’єктно-орієнтований підхід мають на меті допомогти програмістам написати чіткий логічний код для малих та великих проектів.

Python динамічно набирається та збирається сміття. Він підтримує кілька парадигм програмування, включаючи структуроване (зокрема, процедурне), об'єктно-орієнтоване та функціональне програмування. Python часто описують як мову, що включає батареї, завдяки своїй повній стандартній бібліотеці.

Гідо ван Россум почав працювати над Python наприкінці 1980-х років, як наступник мови програмування ABC, і вперше випустив його в 1991 році під назвою Python 0.9.0. Python 2.0 був випущений в 2000 році і представив нові функції, такі як розуміння списків та система збору сміття за допомогою підрахунку посилань. Python 3.0 був випущений в 2008 році і був серйозним переглядом мови, яка не є повністю сумісною із зворотною суттю, і більша

частина коду Python 2 не працює незміненою на Python 3. Python 2 було припинено з версією 2.7.18 у 2020 році.

Python стабільно входить до числа найпопулярніших мов програмування

3.2. Огляд існуючих програмних рішень

Speedtest – браузерне рішення для вимірювання швидкості підключення. Перевагою є простота у використанні, але результати можуть мати похибку залежно від версії браузера, встановлених розширень тощо. Також, варто зауважити, що даний сервіс не призначений для вимірювання швидкості в системі TOR, хоча використовувати його з цією метою можливо.

2ip – браузерне рішення, що не поступається найпопулярнішому, описаному раніше. Із переваг – дане рішення надає можливість заміру швидкості при приєднанні до будь-якої країни світу на вибір, що дає змогу отримати більш повну картину справжньої швидкості з’єднання.

FlashFlow – програмне рішення, що працює як Flash розширення до браузера TOR. Це дає змогу оцінити швидкість підключення у досить повному об’ємі, але технологія, на якій працює додаток, є застарілою. До того ж, використання браузерних розширень у мережі TOR не є безпечним.

3.3. Функціональні вимоги до програмного забезпечення

Програмне забезпечення дозволяє користувачу заміряти швидкість роботи протоколу TOR. Програмне забезпечення має бути призначене до виконання на персональних комп’ютерах з операційною системою Windows версій XP, 7 та 10.

Інтерфейс програмного забезпечення має відображати результати виконання вимірів у графічному вигляді та дозволяти користувачу керувати даними та аналізувати їх. Для цього, результати вимірів повинні зберігатися до файлу формату .csv.

Програмне забезпечення має реалізувати архітектуру MVC. Для розробки додатку була обрана трирівнева архітектура. Трирівнева архітектура - архітектурна модель програмного комплексу, що передбачає наявність у ньому трьох компонентів: клієнта, сервера додатків (до якого підключено клієнтську програму) і сервера баз даних (з яким працює сервер додатків). Клієнт - це інтерфейсний компонент комплексу, надається кінцевому користувачу. Цей рівень не повинен мати прямих зв'язків з базою даних (за вимогами безпеки і масштабованості), бути навантаженим основний бізнес-логікою (за вимогами масштабованості) і зберігати стан додатки (за вимогами надійності). На цей рівень зазвичай виноситься тільки найпростіша бізнес-логіка: інтерфейс авторизації, алгоритми шифрування, перевірка значень, що вводяться, на допустимість і відповідність формату, нескладні операції з даними (сортування, угруповання, підрахунок значень), вже завантаженими на термінал.

Сервер додатків розташовується на другому рівні, на ньому зосереджена велика частина бізнес-логіки. Поза ним залишаються тільки фрагменти, що експортуються на клієнта, а також елементи логіки, занурені в базу даних. Сервери додатків проектуються таким чином, щоб додавання до них додаткових модулів забезпечувало горизонтальне масштабування продуктивності програмного комплексу і не вимагало внесення змін до програмного коду.

Сервер баз даних забезпечує зберігання даних і виноситься на окремий рівень, реалізується, як правило, засобами систем управління базами даних, підключення до цього компоненту забезпечується тільки з рівня сервера додатків.

Відповідно до трирівневої архітектури було обрано шаблон проектування Модель-Вигляд-Контролер.

3.4. Використані зовнішні бібліотеки

Під час розробки будь-якого програмного продукту, часто виникає необхідність використати той чи інший алгоритм, який вже написано та

вдосконалено раніше. Для цього використовують бібліотеки – готові набори функцій та класів для роботи з тою чи іншою предметною областю. В даній роботі було використано наступні бібліотеки:

- configparser - цей модуль забезпечує клас ConfigParser, який реалізує базову мову конфігурації, що забезпечує структуру, подібну до тієї, що міститься у файлах INI Microsoft Windows. Бібліотеку використано це для написання програми Python, яку кінцеві користувачі можуть легко налаштувати;
- multiprocessing - це пакет, який підтримує створення процесів за допомогою API, подібного модулю потоків. Пакет багатопроцесорної обробки пропонує як локальну, так і віддалену паралельність, ефективно побічно застосовуючи Global Interpreter Lock, використовуючи підпроцеси замість потоків. Завдяки цьому багатопроцесорний модуль дозволяє програмісту повністю використовувати кілька процесорів на певній машині. Він працює як на Unix, так і на Windows;
- stem - це бібліотека контролера Python для Tor. Вона дозволяє використовувати протокол управління Tor для створення сценарію процесу Tor;
- flask - мікрофреймворк для веб-додатків, створений з використанням Python. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2.

3.5. Структура програмного продукту

Розроблена програма має два основних файли: main.py та server.py. Файл main.py містить функції із підключення до мережі, надсилання та завантаження файлів на сервер, а також метод для розпізнання файлу конфігурації програми. Наведемо фрагмент файлу для читання файла конфігурації (рис. 3.1.)

```
    config = ConfigParser()
config.read('config.ini')
```

```
ControlPort = int(config['Tor']['ControlPort'])
ServerPort = int(config['Tor']['ServerPort'])
password = config['Tor']['ServerPassword']
url = f'http://www.{config["Tor"]["Url"]}'
```

Рис. 3.1 читання файла конфігурації.

Алгоритм заміру швидкості досить простий: величина файлу у байтах ділиться на час, за який було успішно здійснено його відправку або завантаження (рис. 3.2.)

```
    def get_speed(size: int, duration: float) -> float:
    return size * 8 / 1024 / duration
```

Рис. 3.2 алгоритм визначення швидкості підключення.

Для того, щоб коректно виконувати відправку та завантаження файлу, використовуються користувацькі функції get та post, що відповідають однойменним функціям протоколу http.

Протокол передачі гіпертексту (HTTP) - це протокол прикладного рівня для розподілених спільних інформаційних систем гіпермедіа. HTTP є основою передачі даних для Всесвітньої павутини, де гіпертекстові документи містять гіперпосилання на інші ресурси, до яких користувач може легко отримати доступ, наприклад, клацанням миші або натисканням екрана у веб-браузері.

Розробка HTTP була ініційована Тімом Бернерсом-Лі в CERN в 1989 році. Розробка ранніх запитів HTTP на коментарі (RFC) була скоординованою роботою Групи Інтернет-інженерії (IETF) та Консорціуму Всесвітньої павутини (W3C), що працюють пізніше перехід до IETF.

HTTP метод GET дозволяє отримувати інформацію з HTTP сервера. Інформація, що отримується від сервера може бути будь-якою, головне, щоб вона була в формі HTTP об'єкта. Доступ до інформації при використанні методу GET здійснюється через URI. Часто буває так, що HTTP метод GET звертається до якогось коду, а не до конкретної сторінки (всі CMS генерують контент нальоту), тому метод GET працює так, що ми отримуємо не вихідний код, який генерує текст, а сам текст.

HTTP метод POST є другим по використанню в Інтернеті і потрібен для того, щоб відправляти дані на сервер. HTTP метод POST дозволяє відправляти дані на сервер.

Те, як буде працювати метод POST визначається виключно на стороні сервера і зазвичай залежить від запитуваної URI. Якщо порівняти URI, якого звертається клієнт і повідомлення, яке він хоче відправити з файлової системою, то URI - це папка, а повідомлення клієнта - це файл, який лежить в папці.

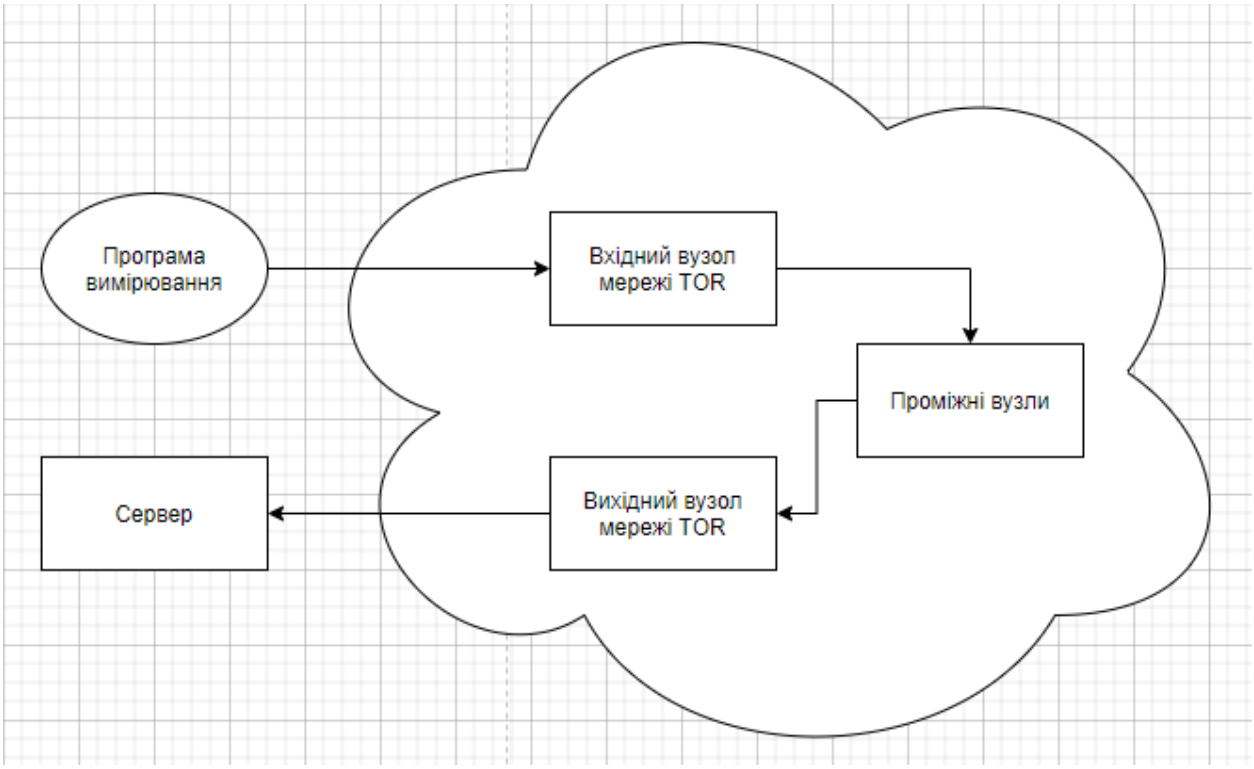
Наведемо методи, що перевизначають методи інтернет-протоколу у розробленому програмному продукті (рис. 3.3.)

```
    def post(session):
    try:
        start_time = time()
        session.get(
            f'{url}/post', data=file_info)
        end_time = time()

        return get_speed(getsize('media/image.gif'), end_time - start_time)
    except Exception as e:
        print(e)


def get(session):
    try:
        start_time = time()
        response = session.get(f'{url}/get')
        end_time = time()
        size = int(response.text.split()[0])

        return get_speed(size, end_time - start_time)
    except Exception as e:
        print(e)
        renew_connection()
        return None
```

Рис. 3.3 Перевизначені методи GET та POST

Файл server.py відповідає за створення серверу, що буде використовувати підключення до мережі TOR. Сервер потрібен для того, щоб запис швидкості

передачі файлів відбувався синхронно із системою вимірювання. До того ж, у мережі TOR досить мало відкритих серверів, що дозволяють отримувати інформацію про їхню роботу, тому було вирішено кожного разу запускати



власний.

Варто зауважити, що хоча власне програма та сервер знаходяться в одній мережі, це не покращує результати випробувань у порівнянні з підключенням до справжнього сервера у мережі TOR. Для наочності, на рис. 3.4. показано алгоритм підключення до власного сервера через мережу TOR.

Рис. 3.4 Мережева карта підключення до сервера

Результати роботи програмного продукту записуються у файл формату .csv, згідно з функціональними вимогами до програми. CSV (від англ. Comma-Separated Values - значення, розділені комами) - текстовий формат, призначений для представлення табличних даних. Рядок таблиці відповідає рядку тексту, яка містить одне або кілька полів, розділених комами.

Формат CSV стандартизований в повному обсязі. Ідея використовувати коми для розділення полів очевидна, але при такому підході виникають проблеми, якщо вихідні табличні дані містять коми або переведення рядків.

Можливим вирішенням проблеми ком і переносів рядків є висновок даних в лапки, проте вихідні дані можуть містити лапки. Крім цього терміном «CSV» можуть позначатися схожі формати, в яких роздільником є символ табуляції (TSV) або крапка з комою. Багато додатків, які працюють з форматом CSV, дозволяють вибирати символ роздільник і символ лапок.

За збереження файлу відповідає метод save_at_csv. Фрагмент коду, що відповідає за збереження результатів, наведено на рис. 3.5.

```
def save_at_csv(con_type, speed, time, date):
    csv_ = []
    with open('excel/speed.csv', newline='') as File:
        reader = csv.reader(File)
        for row in reader:
            if row:
                csv_.append(row)
    with open('excel/speed.csv', 'w', newline='') as f:
        writer = csv.writer(f)
        writer.writerows(csv_)
        writer.writerow([con_type, date, time, speed])
```

Рис. 3.5 Збереження результатів тестування

3.6. Інструкція до використання ПЗ

В папці source знаходиться файли з розширенням py – це вихідний код програмного додатку. В папці media зберігається одна картинка розміром в 1мб. В папці excel знаходиться csv таблиця для запису результатів за наступними правилами: під type пишеться тип швидкості (завантаження з сервера - get_speed, вивантаження на сервер - post_speed), під date пишеться дата початку перевірки швидкості. Під time пишеться час початку перевірки і під speed - швидкість в Кб/с.

В папці config прописуються налаштування. Під заголовком Tor прописуються налаштування для підключення до мережі. У блоці scedule прописується розклад такого формату: time = 10:30 2:17 18:00 (час пишеться

через один пробіл, 24 годинний формат). Під блоком Traffic прописується мінімальне і максимальне значення трафіку для перевірки і вказується в Мб.

Для початку роботи з прогамним додаткомпотрібно встановити Tor сервіс і налаштувати його. Відкрити ControlPort 9051, щоб управляти проксі сервером через бібліотеку stem. Прописати конфігурації в файл torrc (ControlPort і порт проксі сервера). Запустити проксі-сервер Tor і переконатися, що він працює. Далі, в конфігурації програми потрібно прописати папку, де буде зберігається url hidden service Tor і інші файли для роботи сервера (бібліотека самостійно додасть всі файли). У момент запуску сервера генерується url, який потрібно записати в конфігураційний файл (в файлі конфігурації вказується ControlPort, порт проксі-сервера Tor, порт hidden Service, пароль для авторизації Tor (якщо є), розклад перевірок (через пробіл в 24 годинному форматі, до наприклад 12:00 16:00 2:00) і мінімальну та максимальну кількість трафіку для перевірки швидкості.

Сервер може відправляти і приймати картинку. При відправленні картинки він відсилає її бінарний код із зазначенням її розміру на початку. При отриманні сервер просто повертає бінарний код картинки.

Далі, в основних файлах програми вказано функції, в яких будуть відправлятися запити на отримання і відправку картинки. В обох функціях відправляється GET запит, адже Tor не підтримує інших запитів. У цих функціях вираховується швидкість підключення через Tor. Обертають ці функції в інші дві, де відбувається прогін запитів n кількість разів, де n - це ціле випадкове число від min трафіку до max трафіку, ділене на розмір картинки + 1, і вираховується середня швидкість підключення (середня швидкість відправки (вивантаження, post_speed) і отримання (завантаження, get_speed). Ці функції автоматично запускаються в двох процесах, щоб тест був паралельним (за допомогою стандартної бібліотеки multiprocessing). Після прогону середня швидкість записується в csv таблицю, де так само вказується дата початку тесту, час початку тесту і тип швидкості (завантаження, вивантаження).

3.7. Аналіз результатів

За результатами роботи програмного продукту впродовж двох діб (рис. 3.6.), що були записані у формат CSV, а згодом – проаналізовані у програмі Excel, можна зробити наступні висновки:

- Швидкість передачі даних мережею TOR не є стабільною та коливалася у проміжку від 2020,36 Кб/с до 541,81 Кб/с
- Швидкість отримання файлу з сервера методом GET завжди менша за швидкість завантаження файла на сервер методом POST
- Середня швидкість підключення до мережі у 1192,65 Кб/с є досить низькою у порівнянні зі швидкістю підключення звичайною мережею, але достатньою для користування добре оптимізованими веб-ресурсами.

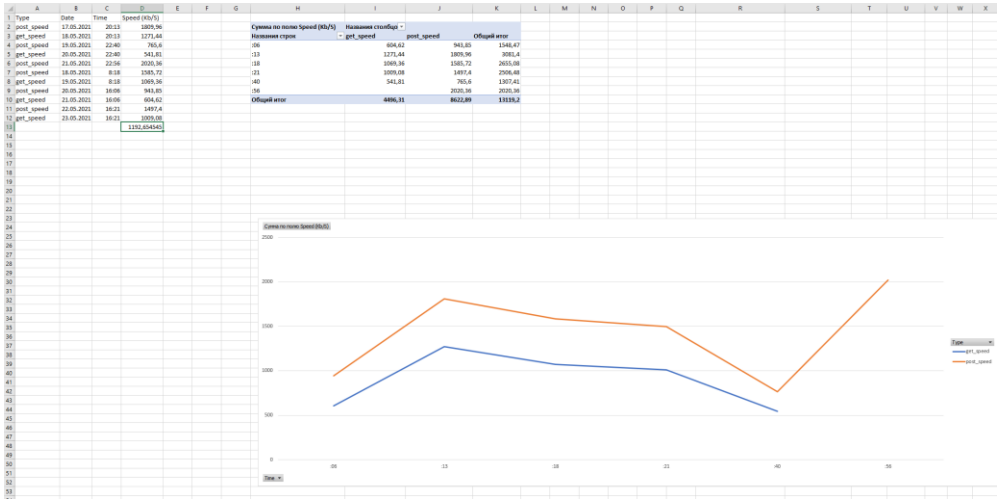


Рис. 3.6 Результати роботи програмного продукту

ВИСНОВКИ

У даній роботі було розглянуто походження та ідею мережі TOR, технологічний огляд цибулевого управління та еволюція технологій протягом трьох поколінь.

Наведено практичну реалізацію використання перенаправлення цибулі на ім'я Tor. Проаналізовано його властивості, алгоритм створення цибулевого ланцюга, проходження повідомлення в ланцюгу цибулі та врахування значення клітин у системі.

Таким чином, програмний додаток цибульного перенаправлення під назвою Tor був створений для забезпечення повної анонімності користувача та його дій в Інтернеті. Використовуючи програмне забезпечення Tor, користувач забезпечує повну анонімність дій в Інтернеті завдяки перенаправленню.

Розглянуто способи та популярні сервіси для виконання вимірювання швидкості підключення до мережі, визначено найбільш підходящий метод для розробки власного ПЗ – виконання вигрузки та завантаження одного й того самого файла на сервер через мережу Tor.

В результаті роботи, розроблено програмний продукт мовою програмування Python. За результатами роботи програмного продукту, здійснено висновок про досить низьку, але достатню для базової роботи, швидкість передачі даних у мережі Tor.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tor: The Second-Generation Onion Router [Електронний ресурс] / R. Dingledine, N. Mathewson, P. Syverson. – 2004. – Режим доступу до ресурсу: https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf.

2. A Peel of Onion [Електронний ресурс] / Paul Syverson. – 2011. – Режим доступу до ресурсу: https://www.acsac.org/2011/program/keynotes/syverson.pdf.

3. Onion routing [Електронний ресурс] – Режим доступу до ресурсу: http://www.onion-router.net/.

4. Tor [Електронний ресурс] – Режим доступу до ресурсу: http://www.torproject.org/.

5. Tor Metrics [Електронний ресурс] – Режим доступу до ресурсу: https://metrics.torproject.org/.

6. Low-Resource Routing Attacks Against Tor [Електронний ресурс] / [K. Bauer, D. McCoy, D. Grunwald та ін.]. – 2007. – Режим доступу до ресурсу: https://www.freehaven.net/anonbib/cache/bauer:wpes2007.pdf.

7. A Stealthy Attack Against Tor Guard Selection [Електронний ресурс] / Q. Li, P. Liu, Z. Qin. – 2015. – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/973a/3ad736c743cb50eacccbfb03e275f8ed2epdf.

8. Recent Attacks On Tor [Електронний ресурс] / Juha Salo. – 2010. – Режим доступу до ресурсу: http://www.cse.hut.fi/en/publications/B/11/papers/salo.pdf.

9. Tor Network Status: TorStatus [Електронний ресурс] – Режим доступу до ресурсу: https://torstatus.blutmagie.de/.

10. Analyzing the Effectiveness of Passive Correlation Attacks on the Tor Anonymity Network [Електронний ресурс] / Sam DeFabbia-Kane. – 2011. – Режим доступу до ресурсу:

https://pdfs.semanticscholar.org/17a6/736b5b8d9a2e516adbebb338e3265681b1c9.pdf.

11. Probabilistic Analysis of Onion Routing in a Black-box Model [Електронний ресурс] / J. Feigenbaum, A. Johnson, P. Syverson. – 2012. – Режим доступу до ресурсу: https://www.ohmygodel.com/publications/wpes08-feigenbaum.pdf.

12. Compromising Anonymity Using Packet Spinning [Електронний ресурс] / [V. Pappas, E. Athanasopoulos, S. Ioannidis та ін.]. – 2008. – Режим доступу до ресурсу: https://www.freehaven.net/anonbib/cache/torspinISC08.pdf.

13. On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records [Електронний ресурс] / [S. Chakravarty, M. V. Barbera, G. Portokalidis та ін.]. – 2014. – Режим доступу до ресурсу: https://www.freehaven.net/anonbib/cache/nfattackpam14.pdf.

14. Low-Cost Traffic Analysis of Tor [Електронний ресурс] / S. J. Murdoch, G. Danezis. – 2005. – Режим доступу до ресурсу: https://www.cs.ucy.ac.cy/courses/EPL682/papers/anon-2.pdf.

15. A New Cell Counter Based Attack Against Tor [Електронний ресурс] / [Z. Ling, J. Luo, W. Yu та ін.]. – 2009. – Режим доступу до ресурсу: http://web.cse.ohio-state.edu/~xuan.3/papers/09_ccs_llyfxj.pdf.

16. Browser-Based Attacks on Tor [Електронний ресурс] / T.Abbott, K. Lai, M. Lieberman, E. Price. – 2007. – Режим доступу до ресурсу: https://www.freehaven.net/anonbib/cache/abbott-pet2007.pdf.

17. A Practical Con-gestion Attack on Tor Using Long Paths [Електронний ресурс] / N. S. Evans, R. Dingledine, C. Grothoff. – 2009. – Режим доступу до ресурсу: https://www.freehaven.net/anonbib/cache/congestionlongpaths.pdf.

18. How Much Anonymity does Network Latency Leak? [Електронний ресурс] / N. Hopper, E. Y. Vasserman, E. Chan-Tin. – 2010. – Режим

доступу до ресурсу: https://www-users.cs.umn.edu/~hoppernj/ccs-latency-leak.pdf.

19. Passive-Logging Attacks Against Anonymous Communications Systems [Електронний ресурс] / M.Wright, M. Adler, B. N. Levine, C. Shields. – 2008. – Режим доступу до ресурсу: http://people.cs.georgetown.edu/~clay/research/pubs/wright-tissec-2008.pdf.

20. AS-awareness in Tor Path Selection [Електронний ресурс] / M. Edman, P. Syverson. – 2009. – Режим доступу до ресурсу: https://www.freehaven.net/anonbib/cache/DBLP:conf/ccs/EdmanS09.pdf.

21. Large Scale Simulation of Tor: Modelling a Global Passive Adversary [Електронний ресурс] / G. Gorman, S. Blott. – 2007. – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/b3e8/7f8d290ac9f38e9321fac7e94c1e74b62e6a.pdf.

22. On the risks of serving whenever you surf: Vulnerabilities in Tor's blocking resistance design [Електронний ресурс] / J. McLachlan, N. Hopper. – 2009. – Режим доступу до ресурсу: https://www.freehaven.net/anonbib/cache/wpes09-bridge-attack.pdf.

23. One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users [Електронний ресурс] / [S. Le Blond, P. Manils, A. Chaabane та ін.]. – 2011. – Режим доступу до ресурсу: https://arxiv.org/pdf/1103.1518.pdf.

24. A potential HTTP-based application-level attack against Tor [Електронний ресурс] / X.Wang, J. Luo, M. Yang, Z. Ling. – 2011. – Режим доступу до ресурсу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.710.6952&rep=rep1&type=pdf.

25. CellFlood: Attacking Tor Onion Routers on the Cheap [Електронний ресурс] / [M. Barbera, V. Kemerlis, V. Pappas та ін.]. – 2013. – Режим

доступу до ресурсу: https://www.freehaven.net/anonbib/cache/esorics13-cellflood.pdf.

26. Peeling back the layers of Tor with EgotisticalGiraffe [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: http://media.encrypted.cc/files/nsa/egotisticalgiraffe-guardian.pdf.

27. Attacking Tor: how the NSA targets users' online anonymity [Електронний ресурс] / Paul Schneier. – 2013. – Режим доступу до ресурсу: https://www.theguardian.com/world/2013/oct/04/tor-attacks-nsa-users-onlineanonymity.

28. The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network [Електронний ресурс] / R.Jansen, F. Tschorsch, A. Johnson, B. Scheuermann. – 2014. – Режим доступу до ресурсу: https://www.freehaven.net/anonbib/cache/sniper14.pdf.

29. New Tor Denial of Service Attacks and Defenses [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://blog.torproject.org/new-tordenial-service-attacks-and-defenses.

30. On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records [Електронний ресурс] / [S. Chakravarty, M. Barbera, G. Portokalidis та ін.]. – 2014. – Режим доступу до ресурсу: https://www.freehaven.net/anonbib/cache/nfattackpam14.pdf.

31. Tor security advisory: "relay early" traffic confirmation attack [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://blog.torproject.org/torsecurity-advisory-relay-early-traffic-confirmation-attack.

32. Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services [Електронний ресурс] / [A. Kwon, M. AlSabah, D. Lazar та ін.]. – 2015. – Режим доступу до ресурсу: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-kwon.pdf.

33. Traffic correlation using netflows [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://blog.torproject.org/traffic-correlation-usingnetflows.

34. Tor suffers traffic confirmation attacks. Say goodbye to anonymity on the Web [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://www.techtimes.com/articles/11711/20140802/tor-suffers-trafficconfirmation-attacks-say-goodbye-to-anonymity-on-the-web.htm.

35. Oracle VM VirtualBox Overview [Електронний ресурс] – Режим доступу до ресурсу: http://www.oracle.com/us/technologies/virtualization/oracle-vmvirtualbox-overview-2981353.pdf.

36. Ferguson N. Cryptography Engineering [Електронний ресурс] / N. Ferguson, B. Schneier, T. Kohno. – 2010. – Режим доступу до ресурсу: http://theeye.eu/public/Books/HumbleBundle/cryptography_engineering_design_principles_and_practical_applications.pdf.

37. Fortuna: Cryptographically Secure Pseudo-Random Number Generation In Software And Hardware [Електронний ресурс] / R.McEvoy, J. Curran, P. Cotter, C. Murphy. – 2006. – Режим доступу до ресурсу: https://www.researchgate.net/publication/215858122_Fortuna_Cryptographically_Secure_PseudoRandom_Number_Generation_In_Software_And_Hardware

38. Testing Random Number Generators [Електронний ресурс] / Dan Biebighauser. – 2000. – Режим доступу до ресурсу: http://wwwusers.math.umn.edu/~garrett/students/reu/pRNGs.pdf.

39. STATISTICAL PROPERTIES OF PSEUDORANDOM SEQUENCES [Електронний ресурс] / Ting Gu. – 2016. – Режим доступу до ресурсу: https://uknowledge.uky.edu/cs_etds/44/.

40. Statistical dependence: Beyond Pearson's ρ [Електронний ресурс] / D. Tjøstheim, H. Otneim, B. Støve. – 2018. – Режим доступу до ресурсу: https://arxiv.org/pdf/1809.10455.pdf.

41. How to modify general TCP/IP traffic on the fly with Trudy [Електронний ресурс] / Tomas Susanka. – 2017. – Режим доступу до ресурсу: https://blog.susanka.eu/how-to-modify-general-tcp-ip-traffic-on-the-fly-withtrudy/.

42. Trudy [Електронний ресурс] / Kelby Ludwig. – 2017. – Режим доступу до ресурсу: https://github.com/praetorian-code/trudy.

43. Palat J. Introducing Vagrant [Електронний ресурс] / Jay Palat. – 2012. –Режим доступу до ресурсу: https://www.linuxjournal.com/content/introducing-vagrant.

44. MITM-VM [Електронний ресурс] / Kelby Ludwig. – 2016. – Режим доступу до ресурсу: https://github.com/praetorian-code/mitm-vm.

45. Fortuna [Електронний ресурс] / Jochen Voss. – 2013. – Режим доступу до ресурсу: https://github.com/seehuhn/fortuna

ДОДАДКИ

Додаток 1 – main.py

```
import csv
from configparser import ConfigParser
from datetime import datetime
from multiprocessing import Process
import multiprocessing
from os.path import getsize
from random import randint
from time import time, sleep
import requests
import schedule
from stem import Signal
from stem.control import Controller

config = ConfigParser()
config.read('config.ini')
ControlPort = int(config['Tor']['ControlPort'])
ServerPort = int(config['Tor']['ServerPort'])
password = config['Tor']['ServerPassword']
url = f'http://www.{config["Tor"]["Url"]}'

with open('media/image.gif', 'rb') as f:
    file_info = f.read()
```

```
# Новое подключение к тору
def renew_connection():
    with Controller.from_port(port=ControlPort) as controller:
        controller.authenticate()
        controller.signal(Signal.NEWNYM)


# Получение новой сессии
def get_tor_session():
    sess = requests.session()
    sess.proxies = {'http':  f'socks5h://127.0.0.1:{ServerPort}',
                    'https': f'socks5h://127.0.0.1:{ServerPort}'}
    return sess


# Получение скорости в Kb/s
def get_speed(size: int, duration: float) -> float:
    return size * 8 / 1024 / duration


def post(session):
    try:
        start_time = time()
        session.get(
            f'{url}/post', data=file_info)
        end_time = time()
```

```
            return get_speed(getsize('media/image.gif'), end_time - start_time)
        except Exception as e:
            print(e)


    def get(session):
        try:
            start_time = time()
            response = session.get(f'{url}/get')
            end_time = time()
            size = int(response.text.split()[0])

            return get_speed(size, end_time - start_time)
        except Exception as e:
            print(e)
            renew_connection()
            return None


    def get_connection_speed(traffic_size, sess, time, date):
        file_size = getsize('media/image.gif') / 1000
        print(f'GET REQUESTS: Объем трафика для теста: {traffic_size /
1000}Mb')
        get_speeds = set()
        iters = int(traffic_size / file_size) + 1
```

```
        for i in range(iters):
            get_sp = get(sess)
            if get_sp:
                get_speeds.add(get_sp)
            print(f'GET REQUESTS: Завершено: {i + 1} из {iters}')
        save_at_csv('get_speed', round(sum(get_speeds) / len(get_speeds), 2), time,
date)



    def post_connection_speed(traffic_size, sess, time, date):
        file_size = getsize('media/image.gif') / 1000
        print(f'POST REQUESTS: Объем трафика для теста: {traffic_size /
1000}Mb')
        post_speeds = set()
        iters = int(traffic_size / file_size) + 1
        for i in range(iters):
            post_sp = post(sess)
            if post_sp:
                post_speeds.add(post_sp)
            print(f'POST REQUESTS: Завершено: {i + 1} из {iters}')
        save_at_csv('post_speed', round(sum(post_speeds) / len(post_speeds), 2),
time, date)



    def test(time):
        renew_connection()
```

```
        session = get_tor_session()
        traffic = randint(int(config['Traffic']['Min']), int(config['Traffic']['Max'])) *
1000
        date = str(datetime.today().date())
        Process(target=get_connection_speed, args=(traffic, session, time, date,
)).start()
        Process(target=post_connection_speed, args=(traffic, session, time, date,
)).start()



    def save_at_csv(con_type, speed, time, date):
        csv_ = []
        with open('excel/speed.csv', newline='') as File:
            reader = csv.reader(File)
            for row in reader:
                if row:
                    csv_.append(row)
        with open('excel/speed.csv', 'w', newline='') as f:
            writer = csv.writer(f)
            writer.writerows(csv_)
            writer.writerow([con_type, date, time, speed])



    if __name__ == '__main__':
        multiprocessing.freeze_support()
        try:
```

```
    print('Tor Requests')
    time_list = config['Schedule']['time'].split()
    for time in time_list:
        if len(time.split(':')[0]) == 1:
            time = '0' + time
        schedule.every().day.at(time).do(test, time)
        print('Set test on', time)
    while True:
        schedule.run_pending()
        sleep(1)
except Exception as e:
    print(e)
input()
```

Додаток 2 – server.py

```
import flask
from stem import Signal
from stem.control import Controller
from flask import Flask
from os.path import getsize
from configparser import ConfigParser
import os


config = ConfigParser()
config.read('config.ini')


app = Flask("Tor Speed Test Server")
port = config['Server']['Port']
host = "127.0.0.1"


print(" * Getting controller")
controller = Controller.from_port(address="127.0.0.1",
port=int(config['Tor']['ControlPort']))
try:
    controller.authenticate()
    controller.signal(Signal.NEWNYM)
```

```
            controller.set_options([('HiddenServiceDir', '/Users/yurarudenko/temp'),
('HiddenServicePort', f'80 {host}:{port}')])
            svc_name = open('/Users/yurarudenko/temp' + "/hostname", "r").read().strip()
            print(" * Created host: %s" % svc_name)
            config.set('Tor', 'Url', svc_name)
            with open('config.ini', 'w') as config_file:
                config.write(config_file)

        except Exception as e:
            print(e)

        with open('media/image.gif', 'rb') as f:
            file_info = f.read()



        @app.route('/get')
        def get():
            return f'{getsize("media/image.gif")} {file_info}'



        @app.route('/post')
        def post():
            return flask.request.data

        if __name__ == "__main__":
            app.run()
```