Міністерство освіти і науки України

Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Мартинюк Тимофій Валерійович,
студент групи НРЗ-181

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Система захисту програмного забезпечення за допомогою апаратного токену

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Кібербезпека

Керівник:
Соколов Артем Вікторович,
к.т.н., доцент

Одеса – 2022

Міністерство освіти і науки України
Національний університет університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти перший (бакалаврський)

Спеціальність 125 – Кібербезпека
Освітня програма – Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри КБПЗ

_____

д.т.н., проф. А.А.Кобозєва

____ _____202_р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мартинюку Тимофію Валерійовичу

1. Тема роботи: *Система захисту програмного забезпечення за допомогою апаратного токену*
Керівник роботи: *Соколов А.В. к.т.н., доцент*.

затверджені наказом ректора від „___” _____ 20__р. № _____.

2. Зміст роботи: *аналіз проблемної області, постановка задачі, аналіз недоліків сучасних методів захисту інформації за допомогою апаратного токену, розробка алгоритму захисту програмного забезпечення за допомогою USB-носіїв, оцінка апаратних токенів різноманітних видів.*
3. Перелік графічного матеріалу*: рисунки інтерфейсу програмного продукту.*

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
Охорона праці	Ярова І.А.	Завдання видав	Завдання прийняв
		16.05.2022	09.06.2022

5. Дата видачі завдання “_____”_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз джерел з теми випускної кваліфікаційної роботи*	*18-11-2021*	*виконано*
2	*Розроблення програмного комплексу*	*20-04-2022*	*виконано*
3	*Написання розділу з охорони праці*	*13-06-2022*	*виконано*
4	*Підготовка тексту роботи*	*10-05-2022*	*виконано*
5	*Підготовка презентації та доповіді*	*24-05-2022*	*виконано*
6	*Попередній захист*	*03-06-2022*	*виконано*
7	*Нормоконтроль, рецензування*	*06-06-2022*	*виконано*
8	*Перевірка на плагіат*	*10-06-2022*	*виконано*

Здобувач вищої освіти _____ *Мартинюк Т.В.*

Керівник роботи _____ *Соколов А.В.*

ЗАВДАННЯ

на розробку розділу “Охорона праці”

Мартинюку Тимофію Валерійовичу, група НРЗ-181

Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Тема роботи *Робоче місце спеціаліста з кібербезпеки*

Зміст розділу:

1 Аналіз умов праці і вибір основних заходів виробничої безпеки.

2 Аналіз пожежної безпеки. Вибір заходів та засобів пожежної безпеки.

Керівник роботи
_____ (Соколов А.В)
«____» _____ 2022 р.

Консультант з охорони праці
_____ (Ярова І.А)
«____» _____ 2022 р.

АНОТАЦІЯ

Дипломна робота на тему «Системи захисту програмного забезпечення за допомогою апаратного токена» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 12 рисунків, 1 таблиця, 1 додаток, 37 літературних джерел за переліком посилань. Робота виконана на 54 сторінках загального тексту і 38 сторінках основного тексту.

Основною метою даної дипломної роботи є розробка системи захисту програмного забезпечення за допомогою USB-токена.

Розробка системи базується на аналізі систем захисту ігор.

Створення USB-токена з USB-флешки обумовлена унікальна дешевизна та простота, що дозволяє в умовах воєнного часу швидко та просто організувати захист.

Створений апаратний токен доступний усім і кожному абсолютно безкоштовно (не на комерційній основі). Система відкрита для модифікацій.

АПАРАТНИЙ ТОКЕН, USB-ТОКЕН, НОСІЙ ЕЦП, АУТЕНТИФІКОВАНЕ ШИФРУВАННЯ, СИСТЕМА ЗАХИСТУ ПО, ДИНАМІЧНІ ШИФРИ, PYTHON.

ANNOTATION

Thesis on the topic "Software protection systems using a hardware token" for the first (bachelor's) level of higher education in the specialty 125 Cybersecurity, specialization, educational program: Cybersecurity, contains 12 figures, 1 table, 1 appendix, 37 references at the list of references. The work is performed on 54 pages of general text and 38 pages of main text.

The main purpose of this thesis is to develop a software protection system using a USB token.

The development of the system is based on the analysis of game protection systems.

Creating a USB token from a USB flash drive is due to its unique cheapness and simplicity, which allows you to quickly and easily organize protection in wartime.

The created hardware token is available to everyone absolutely free of charge (not on a commercial basis). The system is open for modifications.

HARDWARE TOKEN, USB TOKEN, EDS CARRIER, AUTHENTIFIED ENCRYPTION, PROTECTION SYSTEM, DYNAMIC CODES, PYTHON.

ЗМІСТ

ВСТУП

Різні кіберзлочини, наприклад витоку інформації про кредитні картки, крадіжка персональних даних, програми-здирники, крадіжка інтелектуальної власності, порушення конфіденційності, відмова в обслуговуванні — ці інциденти інформаційної безпеки, на жаль, нами почали сприйматися як щось звичайне, як зазвичай пограбування або інше [1]. Для подібних атак немає перешкод, постраждалими можуть бути найбільші, найзаможніші та найбільш захищені підприємства: урядові установи, великі роздрібні мережі, фінансові структури, навіть виробники рішень щодо інформаційної безпеки.

Кіберзлочини з кожним роком стають дедалі витонченішими, саме тому постачальники та творці сервісів з віддаленим управлінням повинні безперервно оновлювати методики та політики безпеки. [2] Кількість атак, ціль, яких ідентифікаційні дані, а домагаються вони їх за допомогою таких способів, як фішинг, брутфорс та атаки за словником. Це означає, що автентифікація більше не може використовувати лише паролі. У цій роботі буде описано проблеми такого методу.

У минулому користувачі повинні були запам'ятовувати та при кожному логіні вводити свої паролі для аутентифікації в більшості програм та веб-сервісів. Окрім незручності, це створювало ще й загрози безпеці, оскільки користувачі часто вибирали слабкі паролі та використовували їх у кількох сервісах. Аутентифікація на основі токенів усуває такі проблеми.

Аутентифікація на основі токенів у поєднанні з додатковими механіками аутентифікації може створити складніший бар'єр, щоб захистити звичайних користувачів від шкідливих дій деяких хакерів. Токени можна отримувати тільки з унікального пристрою, який їх створив (наприклад, смартфона або брелока), завдяки чому вони сьогодні стають високоефективною методикою авторизації.

Хоча платформи токенів аутентифікації зробили великий прогрес, загроза частково зберігається. Наприклад, пристрої токенів, що зберігаються в

мобільних пристроях, токени легко використовувати, але вони можуть виявитися доступними через вразливість пристрою.

Однак одні добре, а два - краще, тому завжди потрібно пам'ятати про те, що ніколи не варто покладатися на один спосіб автентифікації. Аутентифікація токенами повинна вважатися лише одним із компонентів стратегії двофакторної або багатофакторної аутентифікації.

Об’єкт дослідження — процеси використання апаратних токенів.

Предмет дослідження — методи захисту програмного забезпечення за допомогою USB-токену.

Дипломний проект складається з вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

1. АНАЛІЗ ПРОБЛЕМИ СИСТЕМИ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ДОМОГОЮ АПАРАТНОГО ТОКЕНА

1.1. Апаратний токен

У світі кіберзлочини стали проблемами світового масштабу. Безпека даних клієнтів та даних самої компанії є пріоритетом для всіх, не лише великих корпорацій. На Всесвітньому економічному форумі директор BI ZONE у сфері кібербезпеки Дмитро Самарцев навів такі цифри. За його словами, у 2018 році світова економіка від різних кіберзлочинів зазнала збитків у розмірі 1,5 трильйона доларів. Різні проникнення в корпоративні мережі використовуючи чужі дані для аутентифікації склало 31% всіх інцидентів кібербезпеки в 2019 році (з 41 686 задокументованих випадків) [3] безпеки користувачів та їх даних. Нині існує безліч різних методів авторизації та аутентифікації, які створені для реалізації надійної стратегії безпеки. Проте, серед таких методів, експерти виділяють як кращу авторизацію, засновану на використанні токенів. [4]

Раніше використовувалася система паролів і серверів, яка завдяки технічному прогресу незабаром була зміщена після появи токена. Досі така система все ще залишається актуальною саме завдяки її простоті та доступності. Традиційні методи забезпечують своїх користувачів можливістю отримувати доступ до даних у будь-який час. Проте чи це ефективно?

При розгляді цієї системи, зазвичай, можна назвати такі принципи, у яких вона базується:

1. Генерація акаунтів – користувачеві необхідно придумати якесь поєднання літер, цифр або будь-яких дозволених системою символів, які стануть логіном і паролем.

2. Реалізація можливості входу на сервер – користувач повинен зберегти цю унікальну комбінацію, щоб надалі використовуватиме доступу до ресурсу (серверу).

3. Реалізація необхідності знову підключитись до сервера, при цьому відбувається авторизація, для підтвердження ідентифікації користувача. Для

цього користувачеві необхідно наново вводити логін та пароль від свого облікового запису.

Крадіжка паролів – це поширений вид злочинів, особливо в цифровому просторі. Люди влаштовані так, що їм складно запам'ятовувати різні комбінації символів, тому часто записують на звичайному папері (стікери на моніторі, запис в блокноті і т.д.). Фахівці в області кібербезпеки рекомендують створювати вкрай складні, а тому надійні паролі. Для цього необхідно використовувати різні складні символи, не лише числа та літери. Також одним із критеріїв є складання пароля не схожого на якесь слово, яке можна взяти інтуїтивно. Ключова особливість складного та надійного пароля – це той пароль, який важко вгадати чи розгадати.

І так для створення сильного пароля дотримуйтесь наступних рекомендацій: [5]

- Пароль не повинен містити ваше ім'я користувача або справжнє ім'я, а також назви компанії, де ви працюєте
- Довжина пароля має бути не менше восьми символів
- Пароль не повинен містити повного слова
- Пароль повинен відрізнятися від паролів, які ви використовували раніше
- Пароль повинен містити малі, великі літери, а також цифри та символи.

Зловмисник, чия мета вкрасти у Вас паролі в першу чергу перевірятиме такі поширені комбінації: «111111», «123456789», «qwerty», тобто найпростіші набори цифр і букв. Приклади сильних паролів, тобто паролів, які програма буде перебирати дуже довго, виглядають так: HSU5-BHJDa, IkRn%Kmbl253NNp, RF3+yTNBU. Наведені раніше паролі методом підбору за допомогою спеціального програмного забезпечення зламуються за сотні тисяч років, що гарантує їхню безпеку. Проте це лише безпека технічного плану. Не варто забувати, що більшість успішних кіберзлочинів були проведені успішно саме

через вплив людського фактора, який, на жаль, також необхідно враховувати при складанні системи безпеки.

Повертаючись до теми особливостей людини [2], а також отримавши чіткі знання, які паролі вважаються надійними, ми приходимо до відповіді, чому користувачам важко запам'ятовувати такі довгі, але надійні паролі. Це пояснює, чому людям легше записати такий пароль на папері або скласти пароль легше, інтуїтивно зрозуміліший. Таким чином користувачам властиво зберігати всі паролі в одному місці (блокнот або записник), використовувати один і той же варіант у декількох місцях, з невеликими модифікаціями, а саме додавання символів або зміна регістру старого пароля, щоб використовувати його в новому місці. Результатом таких "змін" паролі стають схожими, ідентичними. На жаль, таку долю осягають логіни, аналогічно залишаючись однаковими.

Паролі, крім небезпеки крадіжки даних та складності зберігання інформації, також вимагають перевірки справжності сервера, що збільшує навантаження на пам'ять, бо кожен вхід до системи користувач створює запис такої транзакції на комп'ютері, тим самим збільшуючи список недоліків такого методу.

Віруси, соціальна інженерія, фішинг та інші вектори атаки вказують на те, що самі собою паролі недостатні для адекватного захисту. [6] Незалежно від сфери дії комплексу заходів безпеки: віддалені доступи, дані клієнтів або самої компанії, саме крадіжка облікових записів – це та проблема, яку не можна ігнорувати. Ігнорування ризиків може призвести до фінансових втрат, шкоди репутації, крадіжці інтелектуальної власності та комерційної таємниці.

З іншого боку, існує система авторизації токенів.

Авторизація токенів - це система принцип дій відрізняється зі звичним методом авторизації за допомогою паролів. Авторизація токенів відбувається за допомогою вторинної служби, яка перевіряє запит сервера. Після завершення перевірки сервер видає токен та відповідає на запит. Користувач все ще може зберігати один пароль для запам'ятовування, але токен створює іншу форму доступу, яка краще захищена від крадіжки. Сервер не зберігає запису сеансу.

По суті, токен авторизації – це пристрій, призначений для забезпечення інформаційної безпеки користувача, а також може використовуватись для ідентифікації його власника. Часто токен виглядає як невеликий фізичний пристрій, який використовується для спрощення аутентифікації.

1.2. Види токенів та їх порівняння

Як відомо, токени призначені для електронного посвідчення особи. Вони можуть використовуватись як замість пароля, так і разом з ним. У певному сенсі токен - це електронний ключ для доступа до чогось. Варто згадати, що існує термін щодо програмних токенів, які видаються користувачеві після успішної авторизації на якомусь сервері і є ключем для доступа до таких служб. [2][7]

Апаратні токени часто мають невеликі розміри, що дозволяє носити їх у кишені або в гаманці. Часто оформлені у вигляді брелоків. (Рис. 1.1)

Рис.1.1 – Різновиди зовнішніх видів апаратних токенів

Різні апаратні токени можуть бути використані для зберігання криптографічних ключів, таких, як електронний підпис або біометричні дані. Прикладом біометричних даних то, можливо дактилоскопічний візерунок.

Токени з таким функціоналом мають бути оснащені сканером для відбитка пальця.

Деякі апаратні токени мають міні-клавіатуру для введення PIN-коду або просто кнопка виклику процедури генерації і дисплей для виведення згенерованого ключа. Найчастіше токени мають USB-роз'єм, функції RFID або бездротовий інтерфейс Bluetooth для передачі згенерованої послідовності ключів на клієнтську систему.

Під поняттям «токен» можна зустріти інші назви, наприклад, «електронний ключ», «апаратний ключ», «програмно-апаратний ключ», «ключовий носій». "електронний ідентифікатор", "носій ЕЦП" (електронний цифровий підпис). [8]

Варто відзначити, що останнім часом через популярність біткоінов під токеном стали мати на увазі одиницю криптовалюти, проте для того щоб уникати плутанини в даному в даній роботі термінологія токена буде використовуватися як USB-токен або токен авторизації.

Токени авторизації розрізняються за типами: [9]

1. Токени з підключенням. Пристрої, які потрібно підключити фізично. Наприклад: ключі, диски тощо. Використання токена з підключенням встановлює фізичний зв'язок клієнта та сервера, що усуває користувача вводити дані аутентифікації вручну. USB-пристрій або смарт-карта належать до цієї категорії токенів.

Смарт-карти дуже дешеві та містять перевірені механізми безпеки (зазвичай їх використовують фінансові установи як розрахункові карти). На жаль, такі карти наділені обмеженою обчислювальною потужністю через низьке енергоспоживання та вимоги ультратонких форм. (Рис. 1.2)

2. Токени без підключення. Пристрої, які мають ні фізичного, ні логічного підключення до комп'ютера клієнта. Найчастіше, для них не потрібний спеціальний пристрій введення. Натомість використовується

вбудований екран для відображення згенерованих даних аутентифікації, які, своєю чергою, користувач вводить вручну за допомогою клавіатури. Токени без підключення є найбільш поширеним типом токена, який використовується (зазвичай у поєднанні з паролем) у двофакторній автентифікації для онлайн-ідентифікації. Прикладом такого типу токенів токен SecurID від RSA Security у вигляді брелок.

3. Бездротові токени. На відміну від токенів з підключенням, бездротові токени формують логічний зв'язок з комп'ютером клієнта, при цьому фізичне підключення не потрібне. Відсутність необхідності фізичного контакту робить їх зручнішими, ніж токени з підключенням та токени без підключення. Завдяки своїм особливостям цей тип токенів є популярним вибором для систем входу без ключа та електронних платежів, таких як Mobil Speedpass, які використовують RFID, для передачі інформації про аутентифікацію від токена брелока.

На жаль, такі токени мають 2 проблеми:

3.1. RFID мітки можуть бути легко зламані

3.2. Бездротові токени мають відносно короткий термін служби 3-5 років, тоді як USB токени можуть працювати до 10 років

Рис.1.2 - Смарт-карти та USB-токени JaCarta для аутентифікації

У всіх трьох випадках для запуску процесу користувач має щось зробити. Наприклад, ввести пароль або відповісти на запитання. Але навіть коли ці кроки здійснюються без помилок, доступу без токена отримати неможливо.

1.3. Створення USB-токена з USB-флешки

На USB-токенах можуть зберігатися не тільки ЕЦП, але й паролі, друк, цифрові сертифікати, електронні ключі, біометричні дані тощо. Тому токени відмінно підходять для захисту цифрової інформації та безпечної авторизації на пристроях або сервісах, у тому числі для двоетапної, двофакторної або багатофакторної аутентифікації. [10]

Існує радий відмінностей USB-токена від USB-флешки:

- Ключ, що знаходиться на токені, ніколи не залишає пристрій, а дані з флешки можна легко перенести на інший носій.
- Інформацію, що знаходиться на флешці, може прочитати практично будь-яка людина, а на токені – ні. Вона зашифрована.

- Ресурс USB-накопичувача зазвичай обмежений. У той час як USB-токени від перевірених виробників призначені для більш безпечного та довгострокового зберігання облікових даних.
- Токени, на відміну від флешки, захищені пристрої. Тому зробити USB-токен із флешки не вийде.

Як ми можемо переконатися, відмінності справді істотні. Складно не погодиться, що ліцензовані токени, які отримали Міжнародну сертифікацію, мають ударостійкий корпус, спеціальну апаратну начинку, створену на спеціальних верстатах, а також свою службу підтримки та обслуговування перевершують USB-токен, створений зі звичайної флешки. Однак у наступних розділах буде наведено зворотне підтвердження, а також чим саме USB-токен на основі USB-флешки краще, ніж сертифіковані пристрої.

2. ПРОЕКТУВАННЯ СИСТЕМИ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ДОМОГОЮ АПАРАТНОГО ТОКЕНА

2.1. Принципи реалізації системи захисту програмного забезпечення з використанням USB-флешки

Як було сказано вище, USB токени створені різними компаніями оснащені новітніми і прогресивними технологіями захисту інформації. Апаратні токени використовуються в різних сферах, наприклад, для доступу до веб-ресурсів, для доступу до спеціальних програмних облікових записів на різних серверах. Однак, ця робота буде використовувати апаратний токен для захисту даних програмного забезпечення на персональному комп'ютері користувача.

Далі буде розглянуто список переваг USB-токенів та ряд їх відмінностей від usb-флешки, а також чому usb токен не може бути флешкою по пунктах, для того щоб показати переваги використання usb-накопичувача (для зручності розуміння далі буде звертатися до даного поняття як до "Флешка") для захисту програмного забезпечення на персональному комп'ютері користувача.

2.1.1 «Ключ, що знаходиться на токені, ніколи не залишає пристрій, а дані з флешки можна легко перенести на інший носій»

Це є правдою, однак жодної загрози для безпеки даних користувача не існує. Якщо зловмисник вирішить перехопити ключ або скопіювати файл ключа з флешки, він все одно не зможе відкрити зашифрований файл саме через реалізацію особливої функції. Ця функція перевіряє фізичне ID флешки. Цей ID неможливо підробити або скопіювати жодними методами. Таким чином, коли флешка підключається до комп'ютера користувача, програма розпізнає ID флешку (див. 2.3). За умови, що ID підключеної флешки не співпадає з ID флешки, вказаної при конфігурації даних програми, програма, у свою чергу, не буде реагувати на файли даної флешки. Відповідно, якщо файл був перенесений

на іншу флешку або був скопійований, без володіння оригінальної флешки використовувати даний файл (ключ) неможливо.

2.1.2 «Інформацію, що знаходиться на флешці, може прочитати практично будь-яка людина, а на токені – ні. Вона зашифрована»

Як було сказано вище, на флешці знаходиться лише зашифрований файл із ключем від захищеної папки ПЗ, що знаходиться на ПК. Відкривши цей файл, зловмисник не може отримати для себе ніякої користі, бо в самому файлі знаходиться набір символів, зашифрованих за допомогою AES, розшифрувати та використовувати які може лише алгоритм програми.

2.1.3 «Ресурс USB-накопичувача, як правило, обмежений. У той час як токени USB від перевірених виробників призначені для безпечнішого та довгострокового зберігання облікових даних»

При відповіді на аргумент, наведений цим пунктом, буде розкрито ідею та причину створення даної системи захисту. Ключова особливість, за якою даний проект, а саме USB-токен створюється з USB-флешки, - це дешевизна виробництва.

USB-токени, які надаються кваліфікованими компаніями та розробляються на спеціальних заводах з використанням спеціальної апаратури та різних верстатів, пропрієтарного програмного забезпечення та своєї служби підтримки витрачають на це величезні гроші своїх компаній. Відповідно, щоб заповнити витрати на виробництві, ціни на таку продукцію виставляються відповідно високі. Наприклад, за наведеними даними [7] вартість покупки такого пристрою у приватної фірми, залежно від різноманітності функціоналу, наданого цим токеном, буде різнитися ціна від 10 доларів за штуку до 85 доларів.

Вартість даної системи USB флешки обійдеться тільки у вартість самої флешки (при чому не різниці де потенційний користувач буде купувати її) так як алгоритм знаходиться у відкритому доступі та користування надано абсолютно безкоштовно та повсюдно для всіх бажаючих. Так, щоб будь-яка людина може створити з будь-якої USB-флешки свій власний USB-токен. Крім того, будь-який користувач має повне право модифікувати даний алгоритм. Дані операції дозволені задля комерційних цілей.

Також не варто забувати що саме дешевизна даного пристрою та реалізації даної системи захисту забезпечує повторно використання даного методу при втраті токена або даних все тому що можна створити або купити нову флешку таким же чином вибрати новий об'єкт нову програму і так само прив'язати її до захист за допомогою цієї флешки.

Ще однією ключовою особливістю даного методу та його переваги перед звичайними USB токенами різних кваліфікованих брендів полягає в тому, що навіть після того як USB-флешка стає USB-токеном, вона все ще не втрачає властивості як накопичувач інформації: на ній так само можна продовжувати розміщувати різні дані, необхідні для користувача і паралельно використовувати її для доступу до певних програмних продуктів, при цьому не порушуючи дії системи захисту.

Крім того, так як функції флешки все ще не втрачена за допомогою цієї флешки можна самому контролювати кількість додатків, які необхідно захистити, за допомогою апаратного токена. Під цим мається на увазі, що враховуючи механіку роботи системи захисту, яка створює на USB-токені (флешці) один ключ-файл, а решту дій виконує програма, яка встановлює аутентифікацію між флешкою і персональним комп'ютером, де знаходиться програмне забезпечення в зашифрованій папці, виникає можливість створення величезної кількості різних файлів з такими ключами для кожної програми на цьому пристрої (ПК). Завдяки тому, що файли зберігають у собі лише ключ, представлений кількістю бітів, такі файли вкрай легкі. Володіючи цією інформацією, можна зробити висновок, що можна зберігати безліч файлів-

ключів від різних програм, що знаходяться на ПК користувача. Повторюючи процедуру створення такого ключа та установку захисту для папки з даними кожного програмного забезпечення, можна захистити не одну програму використовуючи лише одну флешку, у той час як токен більш вузькоспеціалізована система і може працювати з одним обліковим записом (наприклад, для авторизації на веб-ресурсі чи певного банківського профілю).

2.2 Недоліки системи захисту

На жаль, така система має свої недоліки. Одним з головних серед них переслідує в такій же мірі ліцензовані токени від сторонніх виробників, а саме: у разі втрати USB-токена втрачається можливість доступу до програмного продукту, якщо він був зашифрований за допомогою даного програмно-апаратного засобу. Можна навести аналогію: у разі втрати ключа, ви більше не зможете відкрити необхідні двері і отримати доступ до потрібних вам ресурсів.

Наступним недоліком, крім втрати даних при втраті флешки є крихкість цієї системи. Як було сказано вище, а саме в списку переваг та відмінностей USB-токену від USB-флешки, апаратні токени створені ліцензійними компаніями виробників, мають більш якісні деталі при своєму збиранні, на відміну від USB-флешки, зовнішній захист якої залежить тільки від виробн ика самої флешки. З цього випливає, що якщо ця флешка була виготовлена з дешевих пластмасових матеріалів вона буде набагато тендітніша і схильна до пошкоджень апаратної частини, а внаслідок чого - втрати даних, ніж флешка, виготовлена з металевих сплавів і має більш захищений корпус. Якщо USB-токен від приватних компаній виробляється приблизно однієї якості та надійності до зовнішніх впливів, у тому числі часу, то захищеність флешки може бути як гірше за цей параметр, так і краще.

2.3 Принцип дії алгоритму програми при взаємодії із USB-токеном

Спочатку в цьому додатку використовується дві різні програми: перша програма називається «конфігуратор». Суть цієї програми налаштувати за допомогою виділених користувачем полів шаблону для створення основної програми main. Main.exe - це основна програма, яка буде виконувати більшу частину важливих функцій.

Програма конфігуратора приймає всі вхідні дані, які ввів користувач у поля tkinter-a, зберігає їх і за допомогою Pyinstaller створює і заповнює шаблон програми Main. Pyinstaller перетворює Main на програму, яка готує файли. Вона створює ключ на флешці, шифрує папку тощо.

Програма Main, створена за шаблоном програми конфігуратора, знаходиться в папці dist, яка створюється при компіляції програми main. Так як необхідно, щоб програма була запущена для приведення її в дію, то перш за все користувач повинен після створення даного файлу EXE і успішної компіляції, запустити цей файл, щоб відправити його працювати у фоновому режимі. Коли користувач запустить програму у фоновому режимі, вона запускає функцію USB-Waiter з усіма параметрами, які вказує користувачем через поля конфігуратора. USB-Waiter займається величезним пластом основних процесів програми. USB-Waiter чекає доки підключиться USB-токен. Далі вона порівнює всі пристрої, що підключаються і, якщо це той пристрій, ID якого користувач вбивав у конфігураторі, відповідно ID даної флешки був відображений в Main, то виходить в USB-Waiter визначає за допомогою методу usb.DeviceID, що це флешка користувача і встановлює коннект з пристроєм. Далі файл, в якому знаходиться пароль від зашифрованої папки (архіву), під владою якого розшифровуватиме папку, яка лежить на диску. Потім він запускає потік генератора паролів, кожні 3 хвилини поки флешка підключена, пароль перегенерується завдяки бібліотеці random і зберігається у файл на флешці кожні 3 хвилини, однак цей параметр можна змінювати. Наприклад, поставити не 180 секунд, а наприклад, кожні 30 секунд змінювати і пароль. Далі, у випадку, коли користувач виймає флешку з ПК, останній пароль, який

знаходився в змінній temp_pass, зберігається на флешці у файлі ключа та зашифровується за допомогою хеш функції та використанням інструментів з бібліотеки hashlib, [11] які ми вказували за допомогою пароля, який перекладається в hash функцію, за допомогою якої зашифровується пароль [4]. Пароль, з якого ця функція створюється із послідовності символів, які користувач вводить на початку, коли він вказує пароль від ключа.

Так як папці (архіву) більше нізвідки брати новий згенерований пароль, то останній пароль, який зберігався в temp_pass зберігається і шифрується за допомогою даного пароля за допомогою та за допомогою AES шифрування шифрується папка, а точніше архів [13]. Далі, коли архів було зашифровано, пароль вдаряється з пам'яті. USB-Waiter продовжує чекати, поки підключиться необхідна для користувача флешка [14].

Після того, як USB-токен був вилучений, у справу входить функція Crypter. Ця функція відповідає за архів і зашифровку даного архіву. Всередині цієї функції відбуваються такі зміни: функція Crypter спочатку визначає за допомогою Main, яка папка була закладена користувачем для захисту. Далі бере всі файли, що знаходяться в цій папці та аналізує їх. Всі файли крім EXE поміщаються в архів, а далі за допомогою AES шифрування зашифровуються тим ключем, який знаходився в змінній temp_pass, далі функція Crypter знаходить всі EXE файли, які були в папці програми. Функція Crypter спочатку підраховує кількість EXE файлів, що знаходилися в папці, а також запам'ятовує їхню назву. Потім Crypter копіює заглушку стільки разів, скільки зустрічалася EXE файл у папці. До того ж, привласнює кожній такій заглушці її унікальне ім'я, пов'язане з такою самою назвою оригінального EXE файлу. Така процедура відбувається для того, щоб користувач не зміг запустити EXE файл програмного продукту. Натомість виводилося на екран повідомлення заглушки: «Прохання увімкнути флешку».

Зворотний процес працює подібним чином. Як тільки USB-Waiter виявляє флешку, робить перевірку, він дістає ключ точніше за пароль, який користувач створював у конфігураторі. За допомогою даного пароля він створює функцію

hash, яка розшифровує ключ-файл на токені. Потім програма USB-Waiter забирає з ключа на токені пароль, вона отримує можливість за допомогою алгоритму розшифрувати архів у папці всі файли за допомогою функції decrypt, яка розшифровує назад на свої місця архів, при цьому всі заглушки видаляються.

Візуальне зображення алгоритму преставлено на рисунку 2.1

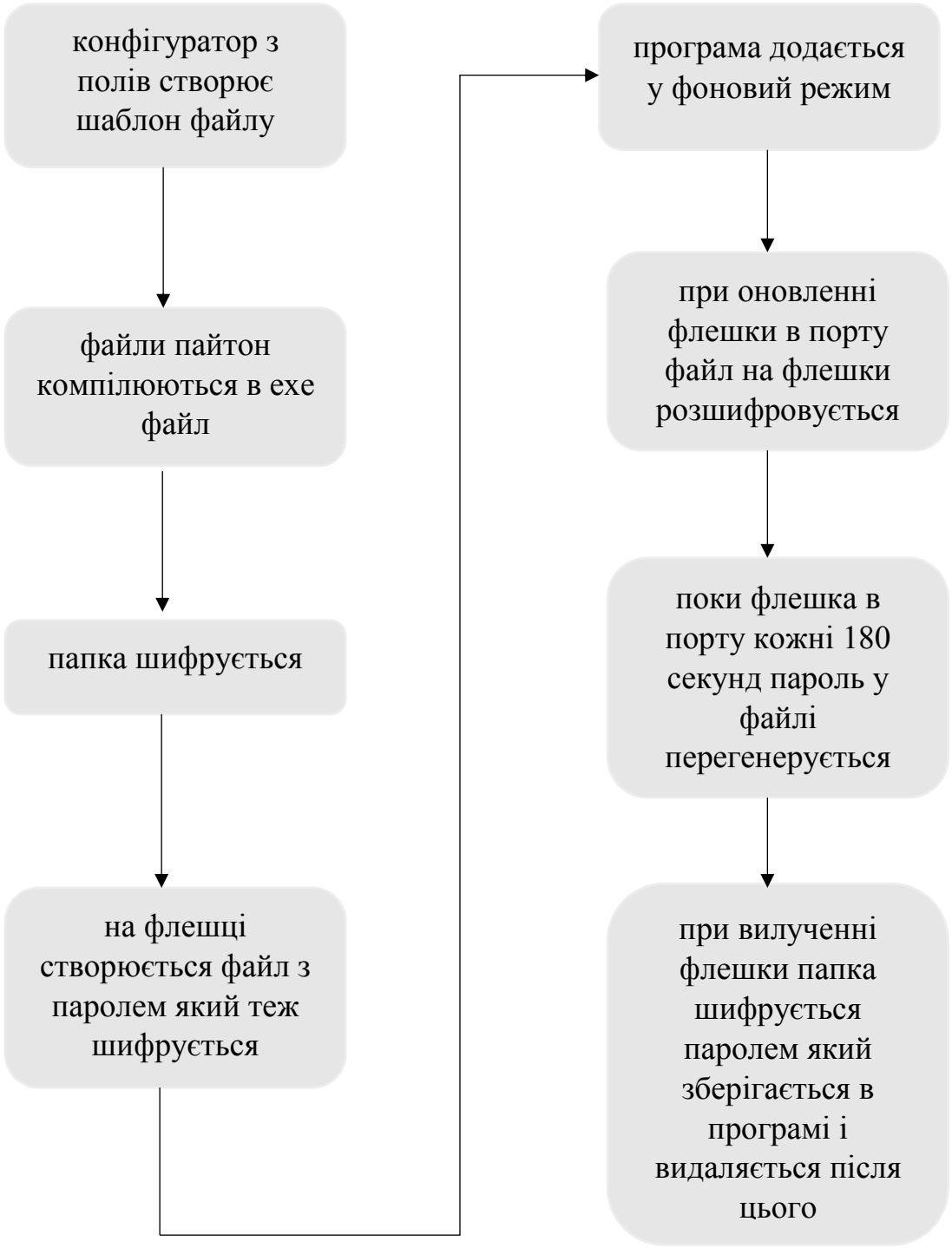

Рисунок 2.1 – Алгоритм послідовності дій програмного комплексу проекту

3. РОЗРОБКА СИСТЕМИ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗА ДОМОГОЮ АПАРАТНОГО ТОКЕНА

3.1. Вибір мови програмування

Python – одна з найпопулярніших інтерактивних мов програмування. [28] Його найбільшою перевагою є простий синтаксис, який дозволяє писати програми з меншою кількістю коду, ніж потрібно для інших мов програмування. А оскільки його синтаксис легко читається та схожий на англійську, всі члени команди можуть легко його зрозуміти

- Python працює на наступних платформах: Windows
- Linux
- macOS

Ви можете виконати код Python одразу після його написання. Однак, оскільки Python є мовою, що інтерпретується, потрібен виконавець - інтерпретатор, що викликає специфічну машину Python. Віртуальна машина Python є важливим компонентом для запуску коду Python. [16]

• Ось як працює програма, написана на Python (рис. 3.1): Вихідний код Python є компілятором, який створює незалежний від ОС проміжний формат файлу, званий байт-кодом Python.

• Віртуальна машина Python інтерпретує байт-код файлу Python для створення спеціальних команд для машини для процесора.

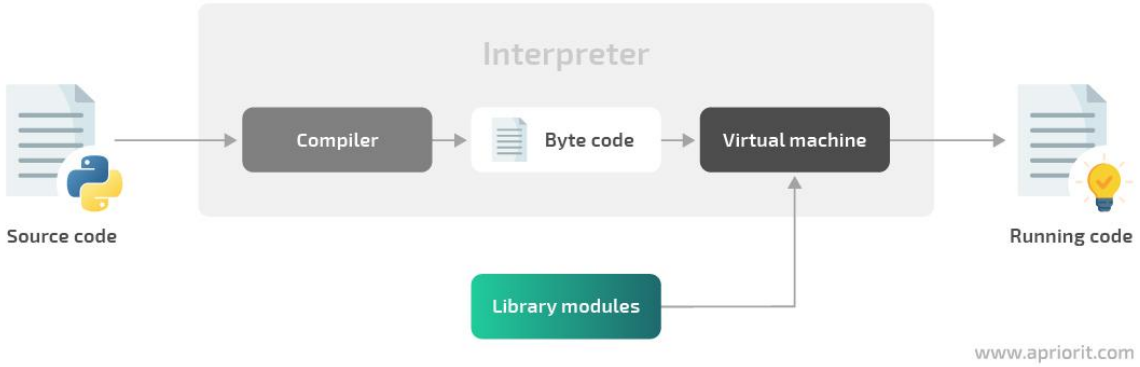

Рис. 3.1 - Схема виконання програми Python

Python має зручні інструменти, такі як pip та віртуальні середовища для обробки залежностей. Крім того, він швидко приймає різні корисні сторонні та пропонує бібліотеки для захисту навколишнього середовища.

Додаткові можливості, такі як математичне моделювання, робота з обладнанням, написання веб-застосунків або розробка ігор можуть реалізовуватися за допомогою великої кількості сторонніх бібліотек.

Наведемо список бібліотек, які були використані для створення цього проекту:

PyCryptodome – це автономний пакет низькорівневих криптографічних примітивів Python.

PyCryptodome — це форк PyCrypto. [17][18]

• Аутентифіковані режими шифрування (GCM, CCM, EAX, SIV, OCB)

• Першокласна підтримка PyPy

• Найкращий і компактніший API

• Алгоритми усіченого хешування SHA-512/224 та SHA-512/256

• Алгоритми хешування BLAKE2b та BLAKE2s

• Потокові шифри Salsa20 та ChaCha20/XChaCha20

• Функції виводу scrypt, bcrypt та HKDF

• Спрощений процес встановлення, включаючи покращену підтримку Windows

• Чистіша генерація ключів RSA та DSA

Далі розглянемо hashlib [11]

hashlib надає хеш-алгоритми SHA-224, SHA-256, SHA-384, SHA-512 на додаток до оптимізованих платформ версій MD5 і SHA1. Якщо є OpenSSL, надаються всі його алгоритми хешування.

Алгоритми хешування - це математичні функції, які перетворюють дані на хеш-значення фіксованого завантаження, хеш-коди або хеш. Вихідне хеш-значення є зведенням вихідного значення. Найбільш важливі у цих хеш-значеннях - неможливість отримання вихідних вхідних даних лише з хеш-значень.

Незважаючи на те, що шифрування важливе для захисту даних (конфіденційність даних), іноді важливо мати можливість знайти, що ніхто не модифікував дані, які ви надсилаєте. значення хешування, яке ви визначаєте, чи не змінилося будь-який файл з моменту його створення (сховища даних).

З hashlib ми використовуємо алгоритм md5: [19]

Алгоритм дайджесту повідомлень MD5 є широко використовуваною хеш-функцією, що виробляє 128-бітове хеш-значення. Його, як і раніше, можна використовувати як контрольну суму для перевірки вмісту даних, але тільки від ненавмисного пошкодження. MD5 перетворює повідомлення про вилучення у вихідні дані фіксованого вилучення з 128 біт. Вхідне повідомлення розбивається на шматки 512-розрядних блоків (шістнадцять 32-розрядних слів); повідомлення доповнюється отже його довжина розкривається на 512.

$$F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$$
$$G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$$
$$H(B,C,D) = B \oplus C \oplus D$$
$$I(B,C,D) = C \oplus (B \vee \neg D)$$

$\oplus, \wedge, \vee, \neg$ захоплюють операції XOR, AND, OR NOT і відповідно.

Наступну бібліотеку, яку ми розглянемо, буде win32api:

Win32 API (також званий Windows API) – це платформа для програм Windows. Цей API найкраще підходить для настільних ігор, потрібен прямий доступ до системних функцій та обладнання. Windows API можна

використовувати у всіх настільних програмах, і ті ж функції зазвичай представлені в 32-розрядній і 64-розрядній версіях Windows.

Важливими функціями, які можна використовувати в win32 Python, є вікна повідомлень, це класичний приклад OK або Cancel.

Python має безліч фреймворків з графічним інтерфейсом, але Tkinter - єдиний фреймворк, вбудований у стандартну бібліотеку Python. Tkinter [20] має кілька сильних сторін. Він кроссплатформенний, тому один і той же код працює у Windows, macOS та Linux. Візуальні елементи візуалізуються з використанням власних елементів операційної системи, тому програми, створені за допомогою Tkinter, виглядають так, ніби вони належать платформі, де вони виконуються.

Tkinter легкий і відносно безболісний у збиранні порівняно з іншими фреймворками. Це робить його переконливим вибір для створення додатків з графічним інтерфейсом на Python, особливо для додатків, де сучасний зовнішній вигляд не потрібен, насамперед пріоритетом є швидке створення чогось функціонального та кроссплатформенного.

Вікно – це екземпляр класу TkinterTk.

Коли ви використовуєте наведений код, на екрані з'явиться нове вікно. Зовнішній вигляд залежить від вашої операційної системи: (рис. 3.2)

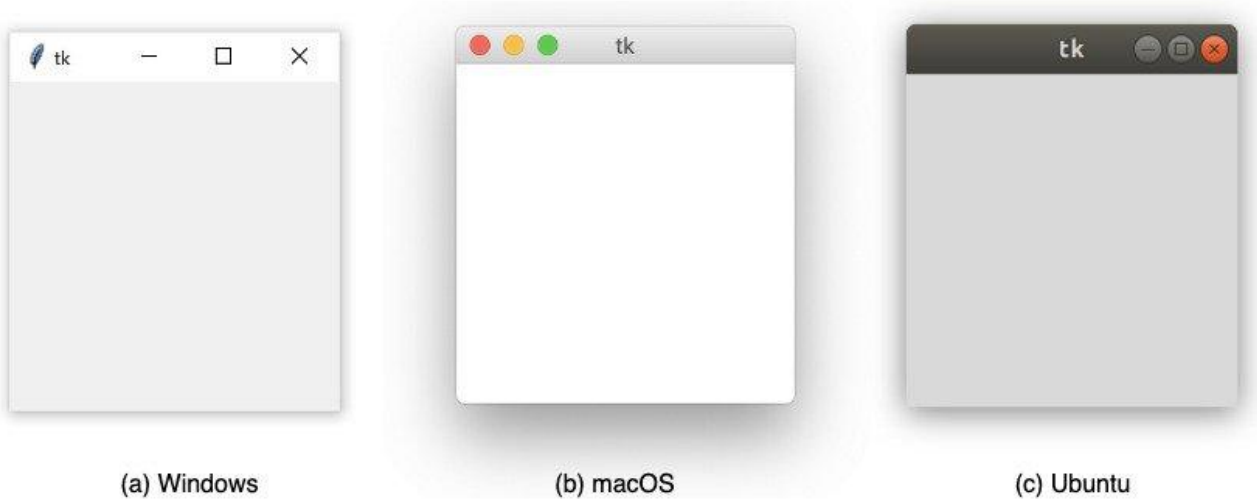

Рис 3.2 – зовнішній вигляд вікна Tkinter залежно від ОС

Завдяки бібліотеці crypter, [21] ми можемо створити та використовувати демон (від англ. слова - daemon) шифрування/дешифрування, який обробляє

запити через сокети домену unix. Демон можна запускати на хості контейнера, а довірені контейнери Docker можуть монтувати каталог сокетів для розшифровки конфіденційних значень конфігурації (тобто облікових даних), не вимагаючи прямого доступу до закритого ключа.

3.2. Опис програми

Щоб реалізувати метод захисту програмного забезпечення за допомогою апаратного токена, нам необхідно створити програму, яка буде виконувати більшість завдань, пов'язаних із захистом даних.

Використання програми починається при створенні файлу main - нашої головної програми, зібраної на основі даних, яких ми отримали з полів у конфігураторі.

Отже, насамперед ми повинні запустити файл конфігуратор (configurator.exe), який виведе на екран невелике вікно завдяки бібліотеці tkinter, яка була описана вище. Дане вікно має кілька полів, які необхідно заповнити даними користувача. (рис. 3.1) Третє поле називається "path to stub.exe". Це поле де ми повинні вказати завдяки методу з бібліотеки Tkinter, за допомогою якого ми можемо вибрати папку за допомогою вікна вибору папок windows [22], директорію, де буде розміщений файл нашої заглушки. Що таке заглушка? Заглушка - це exe файл, чия задача виводити на екран те саме повідомлення (рис. 3). Ця заглушка не має інших призначень, програма створюватиме файл заглушки і перенесе його в певне місце, яке ми вкажемо в цьому третьому полі. Коли наша програма при виконанні свого завдання зашифрує папку з файлами, як відомо всі файли будуть поміщені в зашифрований архів, а на місці файлів які відповідають за запуск і виконання програм будуть створені наші заглушки. Вони приймуть зовнішній вигляд іконки, який був замінений замість цього здійснюваного файлу буде стояти заглушка, яка була просто скопійована n-кількість разів, залежно від кількості exe файлів в цю директорію з тієї директорії, яку ми вказали в третьому полі.

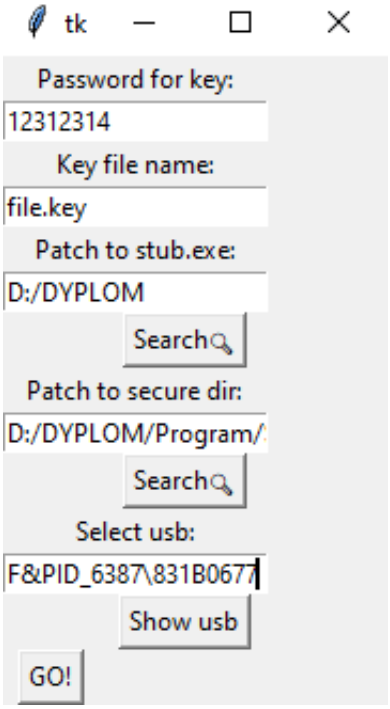

Рисунок 3.1 – Конфігуратор з полями

Четверте поле називається «path to securedir». Це поле відповідає за шлях до директорії, яку нам потрібно шифрувати.

П'яте поле називається "select USB". У цьому полі, користувачеві, необхідно буде вручну розмістити ідентифікаційний номер флешки або пристрою, який буде виступати в ролі USB-токена. Завдяки кнопці «show USB» буде виведено окреме вікно, в якому будуть показані всі підключені зовнішні пристрої та носії інформації, такі як: жорсткі диски, USB-флешки, SSD-накопичувач. (рис. 3.2) Щоб отримати інформацію про нові диски, необхідно натиснути кнопку "refresh". (рис. 3.3) Емпіричним методом вилучення нашої флешки можна дізнатися номер флешки, що використовується. Далі користувачеві необхідно скопіювати цей ID номер у буфер обміну, а заміна вставити його в необхідне поле компілятора і натиснути кнопку «go» що, тим самим запустить компіляцію програми. По завершенню компіляції висвітиться невелике віконце, що позначає успішне завершення операції. Вигляд цього вікна представлений так. (рис. 3.4)

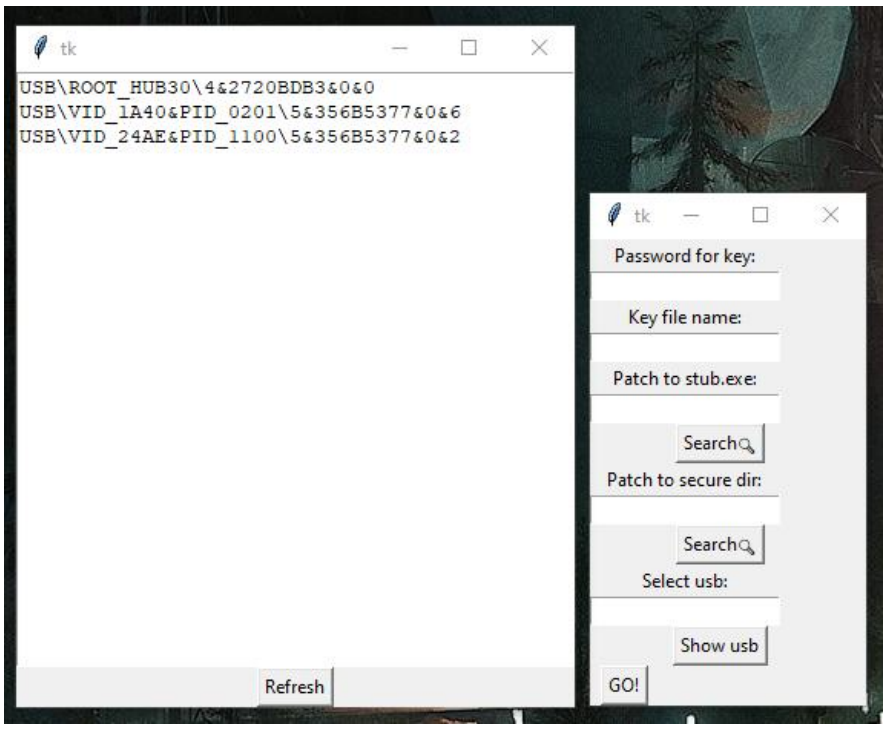

Рис. 3.2 – Натиснення на кнопку «Show usb» - виводить список пристроїв

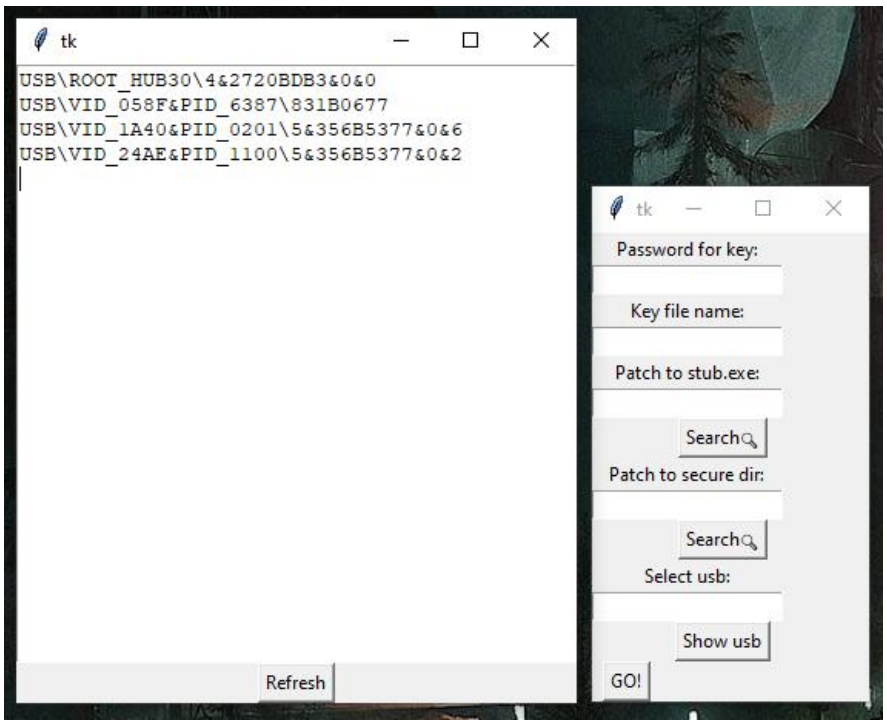

Рис. 3.3 – Натиснення на кнопку «Refresh» - оновлює список пристроїв, тому можна побачити різницю з попереднім слайдом.

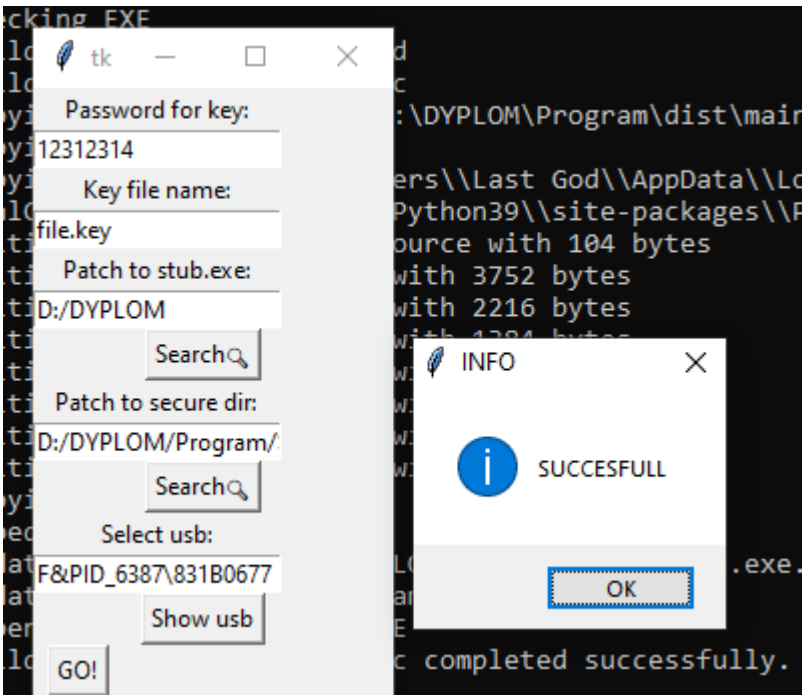

Рис. 3.4 – Повідомлення про успішну компіляцію main.exe

Наступним важливим кроком буде запуск програми. Програма після компіляції знаходиться в папці dist, яка створюється під час компіляції. (рис 3.5) Необхідно двічі клацнути по вказаному файлу main.exe для того, щоб запустити його.

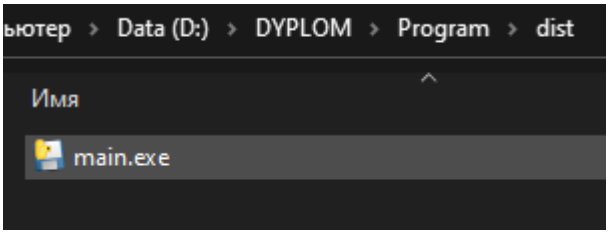

Рис 3.5 – Результат створення програми за параметрами користувача

Після запуску програма буде додана у фоновий режим у комп'ютері. Перевірити це користувач може використовувати Диспетчер завдань (рис. 3.6).

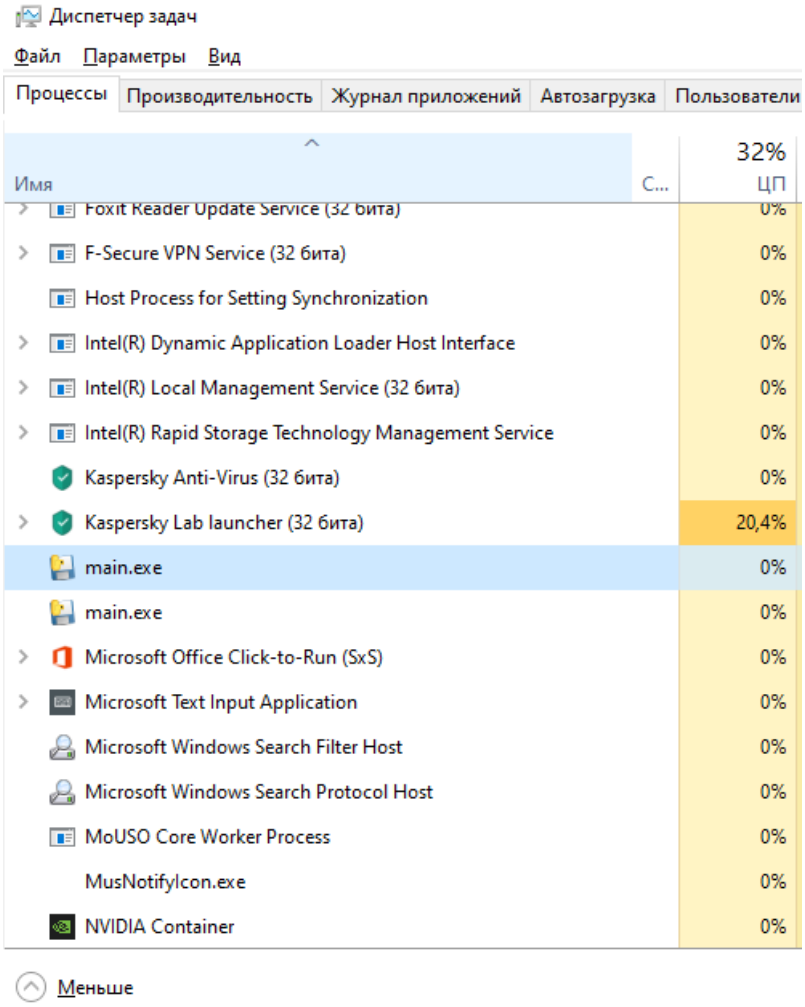

Рис 3.6 – main.exe в диспетчері завдань

Для того, щоб перевірити працездатність програми, необхідно витягти USB-токен з порту USB на стороні комп'ютера. Після цього необхідно перевірити директорію, яку користувач вказує як захисну в четвертому полі конфігуратора. Якщо всередині папки файли заархівувалися, необхідно перевірити можливість доступу до архіву, а саме перевірити, чи вдається користувачеві отримати доступ до архіву. Процедура пройшла успішно, поруч з архівом, у тій же папці, повинні з'явитися заглушки з назвами відповідних для них файлів. Необхідно перевірити, чи виводять вони потрібну для нас інформацію.(рис 3.7 та 3.8)

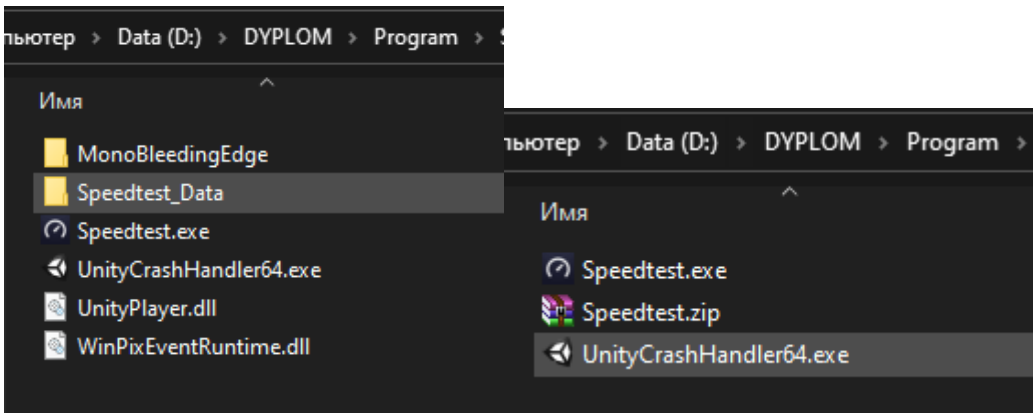

Рис 3.7 – Результат архівації папки програмного забезпечення

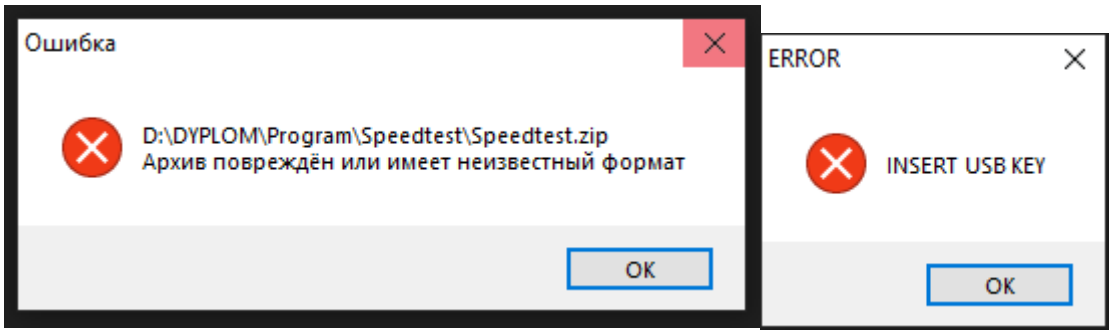

Рис. 3.8 – Показ помилок при намаганні запустити программу після архівації

Наступний та завершальний етап – це перевірка на розшифровку. Необхідно знову підключити USB-токен користувача до ПК через роз'єм USB. Коли комп'ютер визначить USB-пристрій, папка захищена за допомогою апаратного токена повинна буде розшифруватися, з архів виймаються всі дані, які знаходяться в ньому і повертаються в папку, де вони були раніше. Заглушки віддаляються.

Далі слід перевірити працездатність програми: натискаємо файл запуску сторонньої програми. Переконуємось у тому, що програма працює успішно. (рис. 3.9) Якщо всі умови та послідовність завдань була виконана коректно та працездатність програми була перевірена практичним шляхом, можна сміливо вказувати на успішність даного проекту.

4. ОХОРОНА ПРАЦІ

4.1 Аналіз умов праці і вибір основних заходів виробничої безпеки

Завдяки науково-технічному прогресу умови виробничої діяльності робітників чий вид зайнятості передбачає здебільшого розумовий характер було значно змінено. Така праця стала значною, місцями напруженою, але в результаті, як більшість сучасних професій, витратною не тільки в розумовому чи емоційному плані, а й у фізичному. Для вирішення таких проблем необхідно створення безпечних умов праці, ліквідації захворювань, що виникли через професійну діяльність, отримання травм виробничого характеру [23]. Саме сукупність подібних проблем, пов'язаних із забезпеченням здорових та безпечних умов, у яких працює людина, та їх вирішення є одним із головних завдань людського суспільства. Варто звернути увагу, що внаслідок стрімкого розвитку науково-технічного прогресу все частіше знаходить застосування прогресивні моделі наукової організації праці, а також створення кон'єктури, яка здатна виключати професійні захворювання та травматизм. При виявленні всіляких причин нещасних випадків на виробництві, професійних захворювань, аварій, пожеж та вибухів виникає можливість вивчити і створити сприятливі умови для безпечної праці людини, не тільки працюють з ПК, але і будь-якої іншої професії. Як об'єкт проектування для аналізу умов праці розглядається робоче місце спеціаліста з кібербезпеки з персональним комп'ютером (ПК). Це робоче місце складається з письмового столу, оснащеного системним блоком, відеотерміналом (монітор ПК), а також периферія: клавіатура, миша та невід'ємна апаратура офісного робочого місця – принтер та сканер. Наведене вище місце праці, а відповідно його користувач, може бути піддане наступним негативним факторам:

1. Недостатня або підвищена температура повітря;
2. Недостатня чи підвищена вологість повітря;
3. Недостатній рівень освітленості на робочому місці;

4. Підвищений рівень шуму;

5. Підвищений рівень електромагнітного випромінювання;

6. підвищений рівень статичної електрики;

Як і будь-який інший робочий простір, в офісі також може виникнути надзвичайна ситуація, аварія або пожежа, які можуть призвести до вкрай негативних наслідків життєдіяльності людини. Проте, дотримуючись наведених нижче методів охорони здоров'я трудящих, послаблює дії таких наслідків до безпечних меж, а то й зовсім виключає їх.

Як нам відомо, мікроклімат впливає на стан здоров'я, а відповідно на працездатність людини, тому необхідно підтримувати необхідні умови робочої зони в приміщенні [24]. Необхідно дотримуватися значення параметрів мікроклімату, встановлених ДСН 3.3.6-042-99, в якому регламентовано оптимальні та допустимі показники мікроклімату, а також необхідні методи вимірювання мікрокліматичних параметрів для їхньої оцінки. Робота спеціаліста з кібербезпеки за складністю виконуваної роботи відноситься до категорії робіт середньої складності при постійному робочому місці через переважаючий сидячий спосіб робочого процесу, оскільки задіює розумове навантаження. Нормативні дані параметрів з урахуванням згаданих вище умов праці зазначено в таблиці 4.1.

Таблиця 4.1 – Нормування параметрів мікроклімату на робочому місці

Найменування параметра мікроклімату	Період року			
	Холодний		Теплий	
	Оптимальні	Припустимі	Оптимальні	Припустимі
Температура, °С	19 – 21	17 – 24	21 – 23	18 – 27
Відносна вологість, %	40 – 60	75	40 – 60	65 при 26 °С
Швидкість руху повітря, м/с	0,2	0,3	0,3	0,4 – 0,2

Випливаючи з даних, зазначених у таблиці, приходимо до висновку, що температура в холодний і теплий період року повинна бути в межах + 19 – 23 °С, відносна вологість 40 – 60 % за швидкості руху повітря не більше 0,2 м/с. Щоб дотримуватися необхідних параметрів мікроклімату, слід використовувати вентиляційні та обігрівальні системи. Системи опалення та кондиціонування слід встановлювати так, щоб ні тепле, ні холодне повітря не направлялося на людей.

Одне з найважливіших чинників продуктивної роботи для підприємства є раціональне освітлення [25]. Оскільки наш користувач, спеціаліст з кібербезпеки, змушений більшу частину свого робочого часу проводити використовуючи ПК, це безпосередньо впливає на навантаження на органи зору, тому для того, щоб зберегти хороше самопочуття та продуктивну роботу, необхідно створити оптимальний освітлений простір, а саме організувати природние та штучне освітлення робочої зони ПК. Існує безліч досліджень на тему впливу різноманітних освітлення на психологічну складову людини, що безпосередньо впливає на продуктивну і зручну робочу сесію. Наприклад, спектральна гама світла може впливати на настрій робітника. Холодні відтінки сприяють робочому стану людини, а теплі навпаки розслаблюють. Так, наприклад, відомо, що за умови оптимального або підвищеного освітлення робочої зони від 100 лк до 500 лк рівень продуктивності підвищується від 5% до 15%, проте при нестачі світла в приміщенні призводить до напруги зору, послаблює увагу, призводить до передчасної втоми.

Нормативні умови освітлення регламентовані у документі ДБН В.2.5-28-2018. Аналіз цього документа показує, що рівень освітленості повинен бути не меншим за 400 лк.

Робочі місця з ПК найчастіше освітлюють за допомогою змішаного освітлення – природне та штучне. Як штучне освітлення використовується в основному стельові або вбудовані світильники з люмінесцентними лампами. Рекомендовані джерела нейтрально білого світла. При використанні штучних джерел світла необхідно не забувати про концентрацію світла, наприклад, варто

уникати відблисків від різних поверхонь на території робочої зони, таких як монітор або поверхня столу. Тому слід встановлювати джерела з розсіювальним світлом з усіма рекомендаціями, щоб уникнути перерахованих вище проблем. Однак ніяке штучне освітлення не зрівняється з природним освітленням (освітлення приміщень денним світлом). Природне освітлення характеризується тим, що може змінюватися в залежності від часу дня, року, зовнішніх факторів тощо.

Таким чином слід висновок, що нехтувати одним із видів джерела освітлення робочого приміщення не варто. Навіть частка інтесивності шуму може негативно впливати на організм робітника [26]. На жаль, шум оточує нас всюди і протягом усього дня, однак є норми інтенсивності шуму. При вуличному шумі 80-90 дБа, офісний шум в середньому досягає 55-60 дБа. Нормування шуму регулюється щодо ДСП 3.3.6.037-99. За даними документом, норма шуму в офісі на робочих місцях з наявністю ПК не повинна перевищувати 50 дБа. Для досягнення такого рівня шуму при якому на організм робітника буде мінімізовано негативний вплив шуму, слід використовувати звукопоглинаючі матеріали облицювання офісу, а також покращення технічного оснащення прикладних машин для постійного користування на тихіші при роботі, крім цього якісний технічний огляд може покращити якість і термін придатності таких машин, що може добре позначиться витратах зміни технічного забезпечення.

Не варто недооцінювати негативний вплив на слухові органи. При тривалому та інтенсивному впливі шуму на них існує можливість погіршити або зовсім втратити слух. Гучний шум, подібно до освітленості, впливає на концентрацію і продуктивність людини.

Небезпека на людину електромагнітного поля неможливо відчути без спеціального обладнання. Проте, що довше людина перебуваємо у електромагнітному полі, то більше вписувалося шанси поява будь-яких наслідків для організму. Дані різних дослідників свідчать, що при висока частота електромагнітного поля призводить до нагрівання тканин організму.

Крім того, електромагнітні поля впливають на різні системи організму: нервову, імунну, ендокринну та навіть статеву. До такого роду випромінювання чутливість проявляється у таких осіб: діти, вагітні та люди з порушенням серцево-судинної, гормональної та нервової системами.

Щоб убезпечити робочий простір з ПК від електромагнітного випромінювання необхідно не забувати вимикати з мережі прилади та пристрої, що не використовуються, в офісі, стежити за станом техніки і правильно експлуатувати електронні прилади.

Електричні впливи на організм людини, найчастіше, смертельні, тому необхідно дотримуватися правил техніки безпеки на робочому місці під час використання та контакту з електроприладами. [27] Найпоширеніші причини ураження електричним струмом при використанні ПК є:

• Використання електрошнурів при пошкодженій ізоляції;

• Перезавантаження хаба (розетки);

• Використання дешевої електротехніки;

• Рідкісний техогляд та/або ремонт приладів;

• Торкання струмоведучих частин приладів;

• Використання техніки, що перенапружує електромережу, наприклад, прилади для обігріву;

• Використання\зберігання приладів у небезпечному місці, наприклад, біля вологи або вогню;

• Людський фактор, наприклад, неуважність працівника, що може призвести до катастрофічної помилки при використанні електроприладу;

• Не проведення інструктивних подій для підвищення пильності під час експлуатації електроприладів;

Щодо НПАОП 40.1-1.21-98 необхідно мати технічні засоби захисту. Наприклад, заземлення металевих корпусів ПК, що підходять під необхідні параметри напруги, та непошкоджена ізоляція кабелю живлення. Також при невеликих скупченнях ПК на території робочого приміщення необхідно мати

встановлений та налаштований аварійний резервний вимикач повністю всього струму в приміщенні. Вимикач не повинен торкатися функції аварійного освітлення.

4.2Аналіз пожежної безпеки. Вибір заходів та засобів пожежної безпеки

Правила пожежної безпеки в Україні затверджені наказом Міністерства внутрішніх справ України, зі змінами, що періодично вносяться відповідними наказами

Використовуючи дані ДСТУ Б В.1.1-36:2016, можливо визначити організаційні та технічні заходи, які повинен знати та дотримуватися кожен співробітник для запобігання пожежам. Організаційний захід складається ключовим чином із осіб, які пройшли навчання за програмою пожежно-технічного мінімуму, відповідальних за пожежну безпеку у приміщенні, а також за проведення інструктажу для інших працівників підприємства. Таку інформацію, як план евакуації або правила використання вогнегасника, мають бути проговорені на кожному засіданні інструктажу з техніки безпеки під час пожежі, щоб у критично важливий момент аварії не трапилося летальних випадків.

У документі ДСТУ Б В.1.1-36:2016 «позначення категорій приміщень, будівель та зовнішніх установок із вибухопожежної та пожежної небезпеки», є за класифікацією 5 категорії, для офісних приміщень, обране приміщення позначається як категорія «В» - пожежонебезпечною.

Один із головних та обов'язкових елементів документації – це плани евакуації. Такі плани створюються для позначення безпечної процедури евакуації у разі пожежі, де показано основні та запасні маршрути. Такі плани повинні бути легко читані для персоналу підприємства і висіти на доступних для сприйняття відстанях від людини місцях, де їх найкраще видно.

Для успішної ліквідації пожежі необхідно зберігати спокій та не піддаватися паніці. Слід якнайшвидше набрати пожежно-рятувальну службу. Для цього в приміщеннях необхідно встановити електричну пожежну сигналізацію для повідомлення про початок пожежі у перших осередках спалаху. Автоматичні системи пожежної сигналізації є кілька видів, а саме теплові, які реагують на підвищення температури, димові – реагують на дим у приміщенні та змішані чи світлові, які поєднують у собі обидві функції [28].

Висновок

Ця частина дипломної роботи вказує на вимоги робочого простору для фахівця з кібербезпеки. Дотримання вищезгаданих умов та факторів може зберегти відмінну працездатність для всього колективу протягом робочого дня. У цьому зберігши як психологічну, і фізіологічну безпеку людини.

ВИСНОВКИ

У роботі проведено огляд сучасного стану проблеми систем захисту інформації за допомогою апаратних токенів. У ході дослідження було обґрунтовано переваги та недоліки даного методу, а також запропоновано реалізацію створення USB-токену з USB-флешки (flash drive). Як захист інформації виступила програмне забезпечення SpeedTest.

Створення USB-токена з USB-флешки обумовлена унікальна дешевизна та простота, що дозволяє в умовах воєнного часу швидко та просто організувати захист.[29]

Реалізовано власну систему захисту, яка включає такі компоненти:

• блокування файлів

• файл "заглушка"

• динамічний ключ

• конструктор захисту

Створений апаратний токен доступний усім і кожному абсолютно безкоштовно (не на комерційній основі)

ПЕРЕЛІК ПОСИЛАНЬ

1. Рекомендації з інформаційної безпеки для малого та середнього бізнесу (SMB). URL:https://habr.com/ru/post/348892/
2. Токен Авторизації. URL:https://habr.com/ru/post/534092/
3. Обзор аутентификации на основе токенов. URL: https://www.nist.gov/system/files/documents/2019/10/16/1-2-dbir-widup.pdf
4. Огляд аутентифікації на основі токенів. URL: https://habr.com/ru/post/593191/
5. Як створити надійний пароль/ URL: https://imacrosoft.ru/raznoe/kakoj-parol-nadezhnyj.html
6. Мультифакторна аутентифікація (MFA). URL: https://multifactor.dev/files/multifactor.pdf
7. Що таке USB-токен та як його використовувати. URL: https://yubikey.com.ua/ru/chto-takoe-usb-token-y-kak-eho-yspolzovat
8. Вартість різноманітних токенів THE YUBIKEY SERIES 5. URL: https://ipug.com.ua/index.php?route=product/category&path=63_59
9. Security token. *Матеріал з вікіпедії – вільної енциклопедії.* URL:https://en.wikipedia.org/wiki/Security_token
10. Hardware token vs Fingerprint based software token. URL: https://security.stackexchange.com/questions/173529/hardware-token-vs-fingerprint-based-software-token
11. How to Use Hashing Algorithms in Python using hashlib. URL: https://www.thepythoncode.com/article/hashing-functions-in-python-using-hashlib
12. RJ Anderson, “The Classification of Hash Functions”, in 'Codes and Ciphers', proceedings of Fourth IMA Conference on Cryptography and Coding, pp 83–93
13. Daemen, J., Rijmen, V.: The Design of Rijndael. In: AES - The Advanced Encryption Standard. Springer, Berlin (2002)

14. The Hash Function JH 1 - Hongjun Wu 2,3. URL: https://perso.uclouvain.be/fstandae/source_codes/hash_atmel/specs/jh.pdf

15. A Comprehensive Guide to Hooking Windows APIs з Python. URL: https://www.apriorit.com/dev-blog/727-win-guide-to-hooking-windows-apis-with-python

16. Python. *Матеріал з вікіпедії – вільної енциклопедії.* URL: https://ru.wikipedia.org/wiki/Python

17. PyCryptodome AES encryption. URL : https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html

18. Modern modes operation for symmetric block ciphers, URL: https://pycryptodome.readthedocs.io/en/latest/src/cipher/modern.html#eax-mode

19. MD5 message-digest algorithm. URL:https://en.wikipedia.org/wiki/MD5

20. Що таке двоетапна автентифікація. URL: https://yubikey.com.ua/ru/chto-takoe-dvuhjetapnaja-autentifikacija

21. Crypter - Encryption/decryption daemon crypter. URL: https://pypi.org/project/crypter/

22. API Index for desktop Windows applications - Win32 (Windows API). URL: https://docs.microsoft.com/en-us/windows/win32/apiindex/api-index-portal

23. Закон України про охорону праці URL: https://zakon.rada.gov.ua/laws/show/2694-12#top

24. Дослідження мікроклімату у виробничих приміщеннях URL: http://opcb.kpi.ua/wp-content/uploads/2014/09/Микроклимат.pdf

25. Освітлення офісу URL: https://www.optimaltd.com.ua/ua/blog/osveshchenie-ofisa/

26. Наказ про затвердження Державних санітарних норм та правил «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» URL: https://zakon.rada.gov.ua/laws/show/z0472-14#Text

27. Про заходи з електробезпеки для працівників офісу URL: https://oppb.com.ua/articles/pro-zahody-z-elektrobezpeky-dlya-pracivnykiv-ofisu

28. Про затвердження Правил пожежної безпеки в Україні URL: https://zakon.rada.gov.ua/laws/show/z0252-15#Text

29. Закон України про внесення змін до Кримінального кодексу України щодо підвищення ефективності боротьби з кіберзлочинністю в умовах дії воєнного стану URL: https://zakon.rada.gov.ua/laws/show/2149-20/#Text

Додаток А. Лістинг програмного продукту

Файл №1 Configurator.py

```
import win32com.client, win32file, win32api
import win32api
import win32con
from tkinter import *
from tkinter import filedialog
from tkinter import messagebox
from assets.crypter import for_crypt, crypt
import os
import random
import shutil
import hashlib

def add_to_sturtup():
    pass

def crypt_folder(target_dir, pas, patch_stub_app):
    crypt(pas, target_dir, patch_stub_app)



def create_password_file(key_file, key_file_password):
    with open([drive for drive in win32api.GetLogicalDriveStrings().split('\000')[:-1] if win32file.GetDriveType(drive) == win32file.DRIVE_REMOVABLE][-1]+key_file, 'w') as key:
        pas = ''
        for x in range(16):
            pas = pas + random.choice(list('1234567890abcdefghigklmnopqrstuvyxwzABCDEFGHIGKLMNOPQRSTUVYXWZ'))
        key.write(pas)
    for_crypt(key_file_password, [drive for drive in win32api.GetLogicalDriveStrings().split('\000')[:-1] if
```

```
win32file.GetDriveType(drive)    ==    win32file.DRIVE_REMOVABLE][-
1]+key_file)
        return pas

    def replace_stub(patch_stub_app):
        shutil.copyfile('./assets/stub.exe', patch_stub_app)



    def to_exe():
        os.system('pyinstaller --noconfirm --onefile --windowed --
add-data            "./assets/crypter.py;."            --add-data
"./assets/usb_checker.py;."  "./assets/main.py"')



    def     generator(patch_stub_app,     target_dir,     key_file,
key_file_password, usb_hash):
        patter = f"""
    from usb_checker import usb_waiter

    patch_stub_app = '{patch_stub_app}'
    target_dir = '{target_dir}'
    key_file = '{key_file}'
    key_file_password = '{key_file_password}'
    usb_hash        =        '{hashlib.sha256(usb_hash.encode('UTF-
8')).hexdigest()}'
    usb_waiter(target_dir,   key_file,   usb_hash,   patch_stub_app,
key_file_password)"""
        with open('./assets/main.py', 'w') as program:
            program.write(patter)

        to_exe()
        replace_stub(patch_stub_app)
        pas = create_password_file(key_file, key_file_password)
        crypt_folder(target_dir, pas, patch_stub_app)
        add_to_sturtup()
```

```
  def show_usb():
      def refresh():
          usb_list.delete('1.0', END)
          counter = 1
          for dev in [usb.DeviceID for usb in
win32com.client.GetObject("winmgmts:").InstancesOf("Win32_USBHub")
]:
              usb_list.insert(f'{counter}.0', f'{dev}\n')
              counter += 1

      window = Tk()
      usb_list = Text(window, width=45, wrap=WORD)
      usb_list.pack()
      Button(window, text='Refresh', command=lambda:
refresh()).pack()

      window.mainloop()


  def interface():
      root = Tk()

      key_file = StringVar()
      usb_hash = StringVar()
      target_dir = StringVar()
      patch_stub_app = StringVar()
      key_file_password = StringVar()

      Label(root, text='Password for key:').grid(row=0, column=0,
columnspan=2)
      Entry(root, textvariable=key_file_password).grid(row=1,
column=0, columnspan=2)

      Label(root, text='Key file name:').grid(row=2, column=0,
columnspan=2)
```

```
      Entry(root,   textvariable=key_file).grid(row=3,   column=0,
columnspan=2)

      Label(root, text='Path to stub.exe:').grid(row=4, column=0,
columnspan=2)
      Entry(root,         textvariable=patch_stub_app).grid(row=5,
column=0, columnspan=2)
      Button(root,         text='Search ',         command=lambda:
patch_stub_app.set(filedialog.askdirectory())).grid(row=6,
column=1)

      Label(root,   text='Path   to   secure   dir:').grid(row=7,
column=0, columnspan=2)
      Entry(root,  textvariable=target_dir).grid(row=8,  column=0,
columnspan=2)
      Button(root,         text='Search ',         command=lambda:
target_dir.set(filedialog.askdirectory())).grid(row=9, column=1)

      Label(root,   text='Select   usb:').grid(row=10,   column=0,
columnspan=2)
      Entry(root,  textvariable=usb_hash).grid(row=11,  column=0,
columnspan=2)
      Button(root,      text='Show      usb',      command=lambda:
show_usb()).grid(row=12, column=1)

      Button(root,text='GO!',           command=lambda           :
[generator(patch_stub_app.get()+'/stub.exe', target_dir.get()+'/',
key_file.get(),      key_file_password.get(),      usb_hash.get()),
messagebox.showinfo(title='INFO'
,message='SUCCESFULL')]).grid(row=13, column=0, rowspan=2)
      root.mainloop()
```

```
import win32api
import win32con

k=win32api.RegOpenKeyEx(win32con.HKEY_CURRENT_USER,r'SOFTWARE\Microsoft\Windows\CurrentVersion\Run',0, win32con.KEY_ALL_ACCESS)
win32api.RegSetValueEx(k, 'init', 0, win32con.REG_SZ, 'D:\\dist\\module2.exe')

win32api.RegCloseKey(k)
```

Файл №2 Usb_checker.py

```
from crypter import crypt, decrypt, for_crypt, for_decrypt
from threading import Thread
import time
import random
import hashlib
import win32com.client, win32file, win32api

temp_pass = ""
usb_flag = False


def password_generator(patch_to_key, key_file_password):
    global temp_pass, usb_flag
    while usb_flag:
        for_decrypt(key_file_password, patch_to_key)
        with open(patch_to_key, 'w') as key:
            pas = ''
            for x in range(16):
                pas = pas + random.choice(list('1234567890abcdefghigklmnopqrstuvyxwzABCDEFGHIGKLMNOPQRSTUVYXWZ'))
            key.write(pas)
        temp_pass = pas
        print(temp_pass)
```

```
            for_crypt(key_file_password, patch_to_key)
            time.sleep(180)


    def get_password(drive, patch_to_key, key_file_password):
        for_decrypt(key_file_password, drive+patch_to_key)
        with open(drive+patch_to_key, 'r') as file_key:
            key = file_key.read()
        for_crypt(key_file_password, drive+patch_to_key)
        return key


    def usb_waiter(dir_patch, patch_to_key, our_usb, stub_exe_patc,
key_file_password):
        global usb_flag, temp_pass
        while True:
                while our_usb not in
[hashlib.sha256(usb.DeviceID.encode('UTF-8')).hexdigest() for usb
in

win32com.client.GetObject("winmgmts:").InstancesOf("Win32_USBHub")
]:
                    time.sleep(1)
                print('Connected')

                drive_letter = [drive for drive in
win32api.GetLogicalDriveStrings().split('\000')[:-1] if
win32file.GetDriveType(drive)==win32file.DRIVE_REMOVABLE][-1]

                password = get_password(drive_letter, patch_to_key,
key_file_password)

                decrypt(password, dir_patch)

                usb_flag = True
```

```
                Thread(target=password_generator,
  args=(drive_letter+patch_to_key, key_file_password)).start()

                while                    our_usb                    in
  [hashlib.sha256(usb.DeviceID.encode('UTF-8')).hexdigest()  for  usb
  in

  win32com.client.GetObject("winmgmts:").InstancesOf("Win32_USBHub")
  ]:
                    time.sleep(1)

                print('Disconnected')

                usb_flag = False
                password = temp_pass
                crypt(password, dir_patch, stub_exe_patc)
```

Файл №3 Crypter.py

```
from Cryptodome.Cipher import AES
from hashlib import md5

import shutil
import os

password = 'password'


def clear_dir(folder):
    for filename in os.listdir(folder):
        file_path = os.path.join(folder, filename)
        try:
            if              os.path.isfile(file_path)              or
os.path.islink(file_path):
                os.unlink(file_path)
```

```
            elif os.path.isdir(file_path):
                shutil.rmtree(file_path)
        except Exception as e:
            print('Failed to delete %s. Reason: %s' % (file_path,
e))


def archive_create(patch):
    archiv_file  =  shutil.make_archive(base_name=patch.split('/')[-
2], format='zip', root_dir=patch)

    return archiv_file


def unpack(patch):
    shutil.unpack_archive(filename=f"{patch   +   patch.split('/')[-
2]}.zip", extract_dir=patch)


def for_crypt(password, patch):
    zip_file = patch
    password = md5(password.encode('UTF-8')).hexdigest()

    with open(zip_file, 'rb') as file_in:
        file_data = file_in.read()

    key = password.encode('UTF-8')
    cipher = AES.new(key, AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(file_data)

    with open(zip_file, 'wb') as file_out:
        [file_out.write(x) for x in (cipher.nonce, tag, ciphertext)]


def for_decrypt(password, patch):
    zip_file = patch
```

```
    password = md5(password.encode('UTF-8')).hexdigest()

    key = password.encode('UTF-8')

    with open(zip_file, 'rb') as file_in:
        nonce, tag, ciphertext = [file_in.read(x) for x in (16, 16,
-1)]
        cipher = AES.new(key, AES.MODE_EAX, nonce)
        file_data = cipher.decrypt_and_verify(ciphertext, tag)

    with open(zip_file, 'wb') as file_out:
        file_out.write(file_data)


def decrypt(password, patch):
    exe_files = list(filter(lambda file: file.split('.')[-1] ==
'exe', os.listdir(patch)))
    for file in exe_files:
        os.remove(f'{patch}{file}')
    for_decrypt(password, f"{patch + patch.split('/')[-2]}.zip")

    unpack(patch)

    os.remove(f"{patch + patch.split('/')[-2]}.zip")


def crypt(password, patch, stub_patch):
    file_final_patch = patch + patch.split('/')[-2] + '.zip'

    zip_file = archive_create(patch)

    exe_files = list(filter(lambda file: file.split('.')[-1] ==
'exe', os.listdir(patch)))

    clear_dir(patch)
```

```
os.replace(zip_file, file_final_patch)

for file in exe_files:
    shutil.copyfile(stub_patch, f'{patch}{file}')

for_crypt(password, f"{patch + patch.split('/')[-2]}.zip")
```

Файл №4 Main.py

```
from usb_checker import usb_waiter

patch_stub_app = 'D:/DYPLOM/stub.exe'
target_dir = 'D:/DYPLOM/Program/Speedtest/'
key_file = 'file.key'
key_file_password = '12312314'
usb_hash =
'd50c90dec6036ef6e2bf5f96af8b18df4cbc5a7d02f60ee76c5671ece327fc5f'
usb_waiter(target_dir, key_file, usb_hash, patch_stub_app,
key_file_password)
```

Файл №5 Stub.py

```
from tkinter import messagebox
messagebox.showerror(title='ERROR', message="INSERT USB KEY")
```