НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
В.о. завідувача кафедри

_____ **Микола ГРАЙВОРОНСЬКИЙ**
(підпис)

«_____»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною программою «Системи, технології та математичні методи кібербезпеки»
спеціальності: 125 «Кібербезпека»

на тему: Способи виявлення кейлогерів у просторі користувача_____

_____

_____

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи _ФБ-74____
(шифр групи)

Топчій Микита Андрійович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник: Гальчинський Леонід Юрійович_____ _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ - 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Микола ГРАЙВОРОНСЬКИЙ
(підпис)
«___»_____2021 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

_____
(прізвище, ім'я, по батькові)

1. Тема роботи Способи виявлення кейлогерів у просторі користувача___
_____
_____ ,
керівник роботи Гальчинський Леонід Юрійович_____
_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від « » _____ 2021 р. №

2. Термін подання здобувачем вищої освіти роботи 07 червня 2021 р.

3. Вихідні дані до роботи _____
_____
_____

4. Зміст роботи
Вступ
1. Поняття та теоретичні основи про кейлогери
2. Підходи при виявленні кейлогерів у системі
3. Метод виявлення кейлогерів, що не спирається на архітектуру шкідливого програмного забезпечення
Висновки

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)
Презентація

6. Дата видачі завдання 1 грудня 2020_____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Отримання завдання	01.12.2020	виконано
2	Огляд літератури з відповідної теми	05.12.2020	виконано
3	Аналіз сучасних методів виявлення	15.02.2021	виконано
4	Аналіз методів протидії кейлогерам	28.03.2021	виконано
5	Проходження переддипломної практики	12.04 - 16.05.2021	виконано
6	Розробка підходу для виявлення кейлогерів	24.05.2021	виконано
7	Передзахист диплому	03.06.2021	виконано
8	Захист дипломної роботи	16.06.2021	

Здобувач вищої освіти _____ ___Микита Топчій___
(підпис) (Власне ім’я, ПРІЗВИЩЕ)

Керівник роботи _____ _Леонід Гальчинський__
(підпис) (Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота містить: 50 стор., 10 рисунки, 28 джерел.

Актуальність дослідження полягає у тому, що такий тип шкідливого програмного забезпечення як кейлогери з роками не втрачають своєї актуальності, адже є одними з найбільш ефективних способів збору конфіденційнох інформації.

Метою дослідження є обгрунтування та розробка теоретичного підходу при виявленні кейлогерів.

Об'єктом дослідження є принципи функціонування кейлогерів у інформаційних системах та методи їх виявленні

Предметом дослідження є метод вивлення кейлогерів, що функціонують у просторі користувача

Практичне значення результатів полягає у можливості розробки більш гнучкого підходу для виявлення кейлогерів простору користувача

У роботі було розглянуто такий вид шкідливого програмного забезпечення як програмні кейлогери простору користувача. Причиною є те, що даний вид кейлогерів є найбільш розповсюдженим, адже дозволяє ефективно викрадати конфіденційну інформацію. Було проаналізовано методи виявлення кейлогерів простору користувача використовуючи аналіз мережевого трафіку та оперативної пам'яті. У результаті було запропоновано підхід для розробки застосунку для виявлення кейлогерів, надано вказівки щодо використання системних функцій.

БЕЗПЕКА, ШКІДЛИВЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КЕЙЛОГЕРИ, КЕЙЛОГЕРИ ПРОСТОРУ КОРИСТУВАЧА

ABSTRACT

The English abstract must be the exact translation of the Ukrainian “annotation” (including statistical data and keywords).

The relevance of the study is that this type of malware such as keyloggers over the years does not lose its relevance, because they are one of the most effective ways to collect confidential information.

The purpose of the study is to substantiate and develop a theoretical approach to the detection of keyloggers.

The object of research is the principles of functioning of keyloggers in information systems and methods of their detection

The subject of the research is the method of derivation of keyloggers operating in the user's space

The practical significance of the results lies in the possibility of developing a more flexible approach to identifying keyloggers of user space.

The paper considers such a type of malware as software keyloggers of the user's space. The reason is that this type of keyloggers is the most common, because it allows you to effectively steal confidential information. Methods for detecting keyloggers of user space using the analysis of network traffic and RAM were analyzed. As a result, an approach was proposed to develop an application for detecting keyloggers, and instructions were given on the use of system functions.

SECURITY, MALWARE, KEYLOGGERS, USER SPACE KEYLOGGERS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API — Application Programming Interface — інтрефейс, що використовується для визначення поведінки при взаємодії між декількома програмними застосунками

WinAPI — набір базових функцій, інтерфейсів, що надається для взаємодії програмних застосунків та операційної системи

ASCII — American standard code for information interchange — набір таблиць, що використовуються для представлення символів у вигляді числових кодів.

RAM — random access memory — пам'ять з довільним доступом, оперативна пам'ять

АШНК — абстрактний шаблон натискань клавіш — представлення набору символів у вигляді, який буде використаний для подальшого аналізу.

WMI — Windows Management Instrumentation — інструментарій для управління Windows — технологія операційної системи Windows для керування та моніторингом за різноманітними частинами комп'ютерної системи.

GUI — graphical user interface — графічний інттерфейс користувача, спосіб взаємодії користувача із пограмним засобом за допомогою різноманітних графічних елементів.

Драйвер — вид програмного забезпечення, що використовується для взаємодії та управіління периферійних пристроїв операційною системою.

Хук — hook — набір технік, що використовуються для модифікації чи додаванні деякого набору додаткових функції у операційну систему чи програмний застосунок за рахунок перехоплення системних викликів чи повідомлень, що передаються між різними застосунками чи їх

компонентами.

Кейлогер — keylogger — апаратний чи програмний засіб, що використовується для реєстрації різноманітних дій користувачів, зазвичай натискання клавіш на клавіатурі чи моніторинг за переміщеннями курсору мищі. Хоча й використовуються у операційних системах для функціонування комбінацій клавіш, дуже часто є шкідливим програмним забезпеченням, що використовується для зняття конфіденційної інформації.

DLL — Dynamic Link Library — динамічно приєднувана бібліотека

ВСТУП

Внаслідок неперервного розвитку науки людина отримала величезну кількість технологій та пристроїв, що інтегрувалися у наше повсякдене життя для його спрошення та покращення. Однак з ростом різноманітності та складності даних пристроїв виникають багато питань, одним з них є безпека та конфіденційність даних. Це питання має декілька областей, адже під конфіденційною інформацією розуміють як особисті переписки людей, різноманітні файли, так і номери кредитних карт. Тому кіберзлочинці розробили безліч методів отримання конфіденційної інформації, однак мало які методи є такими ж ефективними, як реєстрація натиснених клавіш за допомогою кейлогерів. Такі перехоплені можуть включати вміст документа, паролі, різноманітні ідентифікатори користувача чи номери кредитних карт.

Кейлогери встановлюються на пристрої для моніторингу за діяльністю користувача шляхом фіксації клавіш, що натискаються, та подальшому відправленні даних послідовностей на сервери зловмисників. Зазвичай кейлогери використовуються для викрадення конфіденційної інформації, так викрадається велика кількість паролів, номерів кредитних карт, що ставить даний вид шкідливого програмного забезпечення на місце одних з найбільш небезпечних шпигунських програм.

Традиційні засоби захисту базуються на використанні відбитків, що зазвичай використовуються при виявленні вірусів чи черв'яків. Однак ефективність такої стратегії зменшується при збільшенні різноманіття кейлогерів, через складність постійної підтримки актуальних баз відбитків.

Актуальність дослідження. Актуальність дослідження полягає у тому, що такий тип шкідливого програмного забезпечення як кейлогери з роками не втрачають своєї актуальності, адже є одними з найбільш

ефективних способів збору конфіденційнох інформації.

Метою дослідження є обгрунтування та розробка теоретичного підходу при виявленні кейлогерів. Для розв'язання задачі необхідно вирішити такі завдання:

1) провести огляд опублікованих джерел за тематикою дослідження;

2) класифікувати кейлогери за принципом їх роботи;

3) провести огляд сучасних методів виявлення кейлогерів;

4) розробити підхід для розробки програмного забезпечння для вивлення кейлогерів.

Об'єктом дослідження є принципи функціонування кейлогерів у інформаційних системах та методи їх виявленні.

Предметом дослідження є метод вивлення кейлогерів, що функціонують у просторі користувача.

Практичне значення результатів полягає у можливості розробки більш гнучкого підходу для виявлення кейлогерів простору користувача.

Апробація результатів та публікації. Дана робота була опублікована на XIX Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених "Системи та технології кібернетичної безпеки

1 ПОНЯТТЯ ТА ТЕОРЕТИЧНІ ОСНОВИ ПРО КЕЙЛОГЕРИ

У даному розділі мова піде про види кейлогерів та принципи їх функіонування.

1.1 Принцип роботи клавіатури

Перед початком опису внутрішнього устрою роботи кейлогерів необхідно зазначити основні моменти, що пов'язані з роботою клавіатури.

Клавіатура складається з матриці схем, що пов'язана з клавішами, вона також має назву матриця клавіш. В незалежності від типу даних матриць, коди клавіш, що надсилаються до операційної системи у всіх клавіатурах однакові. На рисунку 1.1 зображено умовну схему функціонування клавіатури.

Коли користувач натискає на клавішу, він замикає схему. Клавіатурний процесор це помічає та визначає позицію клавіші. Після цього процесор переводить позицію на схемі у контрольний код, що надсилається на інтерфейс клавіатури комп'ютера. Клавіатурний контролер отримує цю інформацію та надсилає її до операційної системи, де вже клавіатурний драйвер виконує необхідні дії.

1.2 Види кейлогерів та принципи їх роботи

Зазвичай кейлогери поділяють на дві великі групи: у вигляді апаратних пристроїв та програмного забезпечення. Взагалі кейлогери не

Рисунок 1.1 – Схема роботи клавіатури

завжди є інструментами для зловмисників, що хочуть отримати конфіденційну інформацію, вони можуть використовуються для моніторингом за системою чи при тестуванні програмного забезпечення. Так операційна система та користувацькі застосунки використовують перехоплення клавіш для функціонування комбінацій клавіш (shortcuts), що значно підвищує якість користувацького досвіду. Але в той же час зловмисники використовують кейлогери для отримання конфіденційної інтелектуальної власності, паролів та інших видів інформації. [2][3]

Окрім двох глобальних видів кейлогерів, можна виділити наступні чотири їх категорії:

1) кейлогери у вигляді програмного забезпечення
2) апаратні кейлогери
3) кейлогери для бездротового перехоплення
4) акустичні кейлогери

Апаратні кейлогери

Апаратні кейлогери - це зазвичай електронні схеми, що розташовані між клавіатурою та материнською платою комп'ютера. Існують варіанти встановлення плати кейлогера беспосередньо у клавіатуру, основною перевагою такого підходу є неможливість виявлення закладного пристрою без розбирання клавіатури. В той же час основним недоліком апаратних кейлогерів є необхідність фізичного доступу до цільового пристрою. На рисунку 1.2 можна побачити, який вигляд можуть мати апаратні кейлогери.

Рисунок 1.2 – Приклади апаратних кейлогерів

Принцип роботи апартних кейлогерів наступний:

1) При підключені кейлогеру він починає фіксувати натискання клавіш

2) Для зберігання послідовностей натискання використовується запис на внутрішню пам'ять кейлогера. Даний процес є невидимим для користувача.

3) Після запису інформації на внутрішній накопичувач зловмисник може отримати інформацію за допомогою програмного забезпечення, що надається разом з апартним пристроєм. За наявності бездротових інтерфейсів у кейлогері, таких як Bluetooth, можна отримати записану інформацію знаходячись на невеликій відстані від пристрою жертви.

Таким чином основною перевагою використання апаратних

кейлогерів є їх невидимість для антивірусного програмного забезпечення, однак недоліком є необхідність у фізичному доступі до комп'ютера жертви.

Кейлогери для бездротового перехоплення

З появою та розповсюдженням бездротових клавіатур з'явилась нова загроза конфіденційності інформації. Хоча радіус дії бездротових клавіатур досить невеликий (до 10-ти метрів) цього достатньо, щоб перехопити радіохвилі, які переносять інформацію про натиснені клавіші. Зазвичай сучасні виробники не передають інформацію у відкритому вигляді при передачі радіохвилями, однак те шифрування, що використовується є дуже слабким.[24] Тому виникає проблема, що зловмисник, який знаходиться в радіусі дії бездротового пристрою може перехопити пакети з бездротової клавіатури та легко отримати надіслану інформацію.

Однак однією з основних проблем є необхідність використання антени для перехоплення сигналу близько до жертви.

Акустичні кейлогери

Даний тип кейлогерів потребує спеціального обладнання для прослуховування користувацького вводу та спеціального програмного забезпечення для подальшого аналізу отриманих записів. Кожна клавіша на клавіатурі при натисканні виробляє дещо різні звуки, таким чином використовуючи акустичний аналіз на основі статистичних методів,

можна отримати приблизну послідовність натиснень клавіш. [23] Перевагою такого методу є те, що мікрофони, які використовуються для запису звуків є достатньо чутливими, щоб вловити інформацію на великій відстані.

Програмні кейлогери

Кейлогери у вигляді програмного забезпечення можна додатково класифікувати в залежності від привілеїв, що необхідні для функціонування. Кейлогери, що написані на рівні ядра працюють з повними привілеями. В той же час можлива реалізація у непривілейованому режимі, де кейлогер запускається як звичайний процес простору користувача. Зазвичай кейлогери рівня користувача використовують набори непривілейованих API(Application programming interface – прикладний програмний інтерфейс) доступних на сучасних операціних системах. Даний підхід значно спрощує розробку програмного забезпечення, адже розробка кейлогера на рівні ядра вимагає значно більших зусиль та знань з впровадження та відлагодження. [7][9]

Через складність використання та реалізації кейлогерів, що були описані раніше, саме кейлогери у вигляді програмного забезпечення є найбільш поширеними. Серед них виділяють кейлогери на основі гіпервізора, що являє собою шар між апаратним забезпеченням та операційною системою. Кейлогери ядра зазвичай реалізуються як частина більш складного руткіта, відмінність полягає у тому, що хуки використовуються беспосередньо для перехоплення подій обробки буферу чи інших повідомлень ядра.

Кейлогери рівня ядра ефективні, але їх використання вимагає привілейований доступ до пристрою. [27]В той же час кейлогери простору

користувача не потребують спеціальних прав для розгортання у системі. Їх виконання не залежить від наданих привілеїв. Окрім цього кейлогери простору користувача значно простіші у розробці через добре описані та відлагоджені API.

Кейлогери простору користувача можуть бути класифіковані в залежності від виду повідомлень та структур даних, що перехоплюються. Позаяк на пристрої може бути встановлено декілька застосунків, натиснені клавіші можуть перехоплюватись як глобально, тобто всіма застосунками, так і локально, тобто всередині застосунка. Обидва види кейлогерів можуть бути реалізовані засобами Windows без використання додаткових бібліотек.

Таким чином можна зробити наступні висновки: більшість кейлогерів простору користувача реалізовані на базі хуків клавіш; API операційних систем дозволяють використовуваи ці хуки. Важливим моментом також є те, що ці API дають змогу перехоплювати натискання клавіш у випадку, коли застосунок є не у фокусі. Це викликано необхідністю при використанні клавіатур зі спеціальними клавішами, роботи віконних менеджерів з комбінаціями клавіш системи та виконання команд, що викликаються комбінаціями клавіш, які запрограмував користувач.

Програмні кейлогери поділяють на 4 головні категорії:

– кейлогери на основі циклічного опитування
– кейлогери на основі хуків
– руткіт кейлогери
– кейлогери рівня ядра

Кейлогери на основі циклічного опитування

Даний вид кейлогерів є найпростішим у реалізації та його можна легко виявити. Він використовує набір API функцій, що повертають інформацію про натискання клавіш. Дані функції циклічно опитують стан клавіатури для виявлення натискань, для прикладу було натиснуто клавішу, функція *GetAsyncKeyState* визначає, що саме за клавіша, після цього послідовність

натискань записується до буферу.

Кейлогери на основі хуків

При розробці кейлогерів класичним методом вважається використання хуків (hooks). Даний метод використовується для застосунків з графічним інтерфейсом користувача (Graphical user interface) та дозволяє перехоплювати не тільки натискання клавіш, а й повідомлення, які оброблюються у вікні застосунку. При розробці кейлогерів з використанням хуків, механізм перехоплення повинен бути розміщений у DLL(Dynamic link library – динамічно приєднувана бібліотека), які будуть завантажувати застосунки. Для прикладу функція SetWindowHookEx може бути використана для контролю системи на певні типи подій таких як натискання клавіш.

Руткіт кейлогери

На відміну від кейлогерів на основі хуків руткіт-кейлогери є значно небезпечнішими, однак і зустрічаються відносно рідко. Вони перехоплюють набір функцій, що відповідають за обробку повідомлень чи обробку введеного тексту. Зазвичай при розробці таких кейлогерів застосовують функції GetMessage, TranslateMessage та PeekMessage, що знаходяться у системній біблотеці user32.dll, вони дозволяють впіймати повідомлення та спостерігати за повідомленнями, що отримує застосунок із GUI.

Кейлогери рівня ядра

Велика кількість кейлогерів написані на рівні ядра. Основною технікою є встановлення модифікованого драйвера пристрою введення. Таким чином такі кейлогери заввантажуються разом із системою. Для функціонування такого типу кейлогерів зазвичай користуються IOAttachDevice та IOCreateDevice.

Методика розробки кейлогерів

Зазвичай при розробці кейллгерів дотримуються однієї з трьох методологій:

– метод на основі хуків клавіатури у Windows (Windows keyboard hook)

– метод на основі Таблиці станів клавіатури

– метод на основі спеціального фільтру-драйверу рівня ядра

Перший метод базується на функціях для моніторингу за станом клавіатури на основі хуків, що надає операційна система. Коли натискається клавіша операційна система записує дану подію та реєструє активний застосунок. Пізніше будь-яке повідомлення перехоплюється кейлогером перед тим, як потрапити до пункту призначення. На сьогоднішній момент зазвичай використовується саме даний метод через простоту розробки. На рисунку 1.3 відображено діаграму, що демонструє механізм перехоплення на основі хуків.

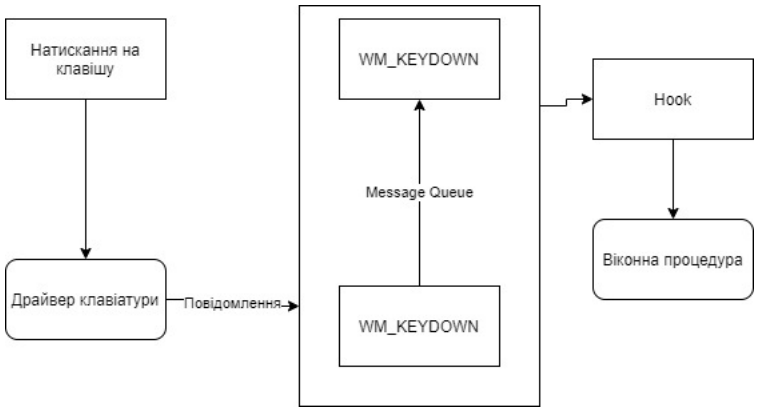

Рисунок 1.3 – Діаграма механізму хуків

Другий метод базується на таблиці станів клавіатури, що містить стан 256 віртуальних клавіш. Застосунки, що використовують віконний інтерфейс посилаються на дану таблицю, вони її використовують, щоб визначити статус клавіші. На рисунку 1.4 зображено схему роботи

кейлогера на основі таблиці станів клавіатури.

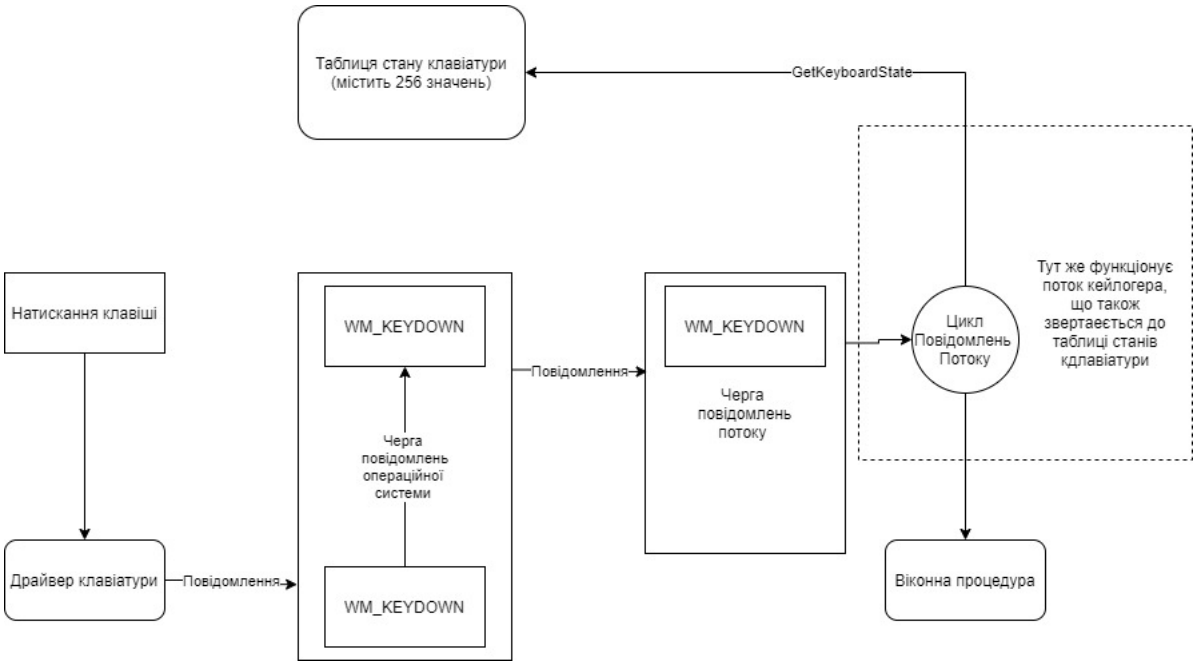

Рисунок 1.4 – Метод на основі таблиці станів клавіатури

Кейлогери, що базуються на фільтрі-драйвері функціонують на рівні ядра, їх важко виявити, однак для встановлення на цільову машину вони потребуть прав адміністратора.

Висновки до розділу 1

У даному розділі було розглянуто основні види кейлогерів та короткі огляди принципів їх роботи. Було наведено перелік типів програмних та апаратних кейлогерів. Додатково було розглянуто методику розробки програмних кейлогерів, що базуються на використанні таблиці станів клавіатури та на основі використання хуків клавіатури. Таким чином було вирішено задачу ознайомлення з основною інформацією про кейлогери. Результатом став висновок, що на даний момент найбільш популярними видами кейлогерів є програмні кейлогери простору користувача, через простоту їх розповсюдження, розробки та відлагодження.

2 ПІДХОДИ ПРИ ВИЯВЛЕННІ КЕЙЛОГЕРІВ У СИСТЕМІ

У даному розділі мова піде про сучасні методи, що використовуються для виявлення кейлогерів у системі та проблеми, з якими зустрічаються дані методи. Окрім цього буде розглянуто метод обходу кейлогера при введені конфіденційної інформації.

2.1 Типи технік, що використовуються при виявленні кейлогерів

Розповсюдженим засобом виявлення кейлогерів є антикейлогери - програмне забезпечення, що слідкує за працюючими застосунками системи використовуючи хуки Windows. Основною проблемою даного програмного забезпечення є велика кількість хибно-позитивних результатів, через застосунки, що використовують перехоплення клавіш у легітимних цілях, для функціонування комбінацій клавіш.

Однією з технік, що використовується для протидії функціонуванню кейлогерів є антихук. Він полягає у тому, щоб для початку знайти усі процеси в системі, що використовують хуки. Існує можливість це зробити внезалежності видимий цей процес чи ні. [1]

Після сканування системи та знаходження процесів, що використовують хуки перелічуються усі динамічні бібліотеки, що при цьому були завантажені. Знаючи, що функції для встановлення хуків знаходиться у певній системній бібліотеці, можна без проблем вирахувати необхідний процес. Так функція *SetWindowHookEx* знаходиться у бібліотеці *USER32dll*. Подальша задача полягає у знаходженні процесу, що використовує дану функцію. Послідовність дій при такому підході

проілюстрована на рисунку 2.1 [13].

Рисунок 2.1 – Послідовність дій виявлення кейлогерів на основі пошуку процесів, що використовують хуки

2.2 Виявлення кейлогерів на основі аналізу мережевого трафіку та пам'яті

Даний підхід може бути використаний при виявленні програмних кейлогерів. За допомогою даного методу потенційно можна виявити будь-який вид кейлогерів, що встановлені в системі. Майже всі види кейлогерів створюють файл логів на комп'ютері-жертві для зберігання натиснутих клавіш, однак існують такі види шкідливого програмного забезпечення за якого лог-файли зберігаються у RAM(Random Access Memory – пам'ять з довільним доступом, оперативна пам'ять) як у буфері. Дана особливість виводить кейлогери на новий рівень та робить їх більш складними для виявлення.[25]

Велика кількість антивірусних засобів має модуль для виявлення кейлогерів, однак основною проблемою цих засобів є підтримка актуальної бази сигнатур. Велика кількість кейлогерів простору користувача з унікальними патернами поведінки з'являються постійно

саме тому метод виявлення кейлогерів на основі відповідності сигнатур є не завжди ефективним. Відомі підходи за яких за допомогою спеціальної утиліти до процесів системи надсилаються згенеровані строки після чого ведеться спостереження за шаблонами зміни розміру пам'яті для цих процесів. Однак недоліком даного підходу є необхідність у спостереженні у реальному часі за цими змінами. Вище описані проблеми можна розв'язати використовуючи інший підхід.

Програмні кейлогери зазвичай для свого маскування використовують руткіти. Зловмисник має змогу вмонтувати кейлогер у будь-який застосунок у системі. В той час, коли кейлогер працює разом з іншим застосунком, то він стає невидимим для менеджеру задач. У випадку роботи віддаленого кейлогера процес буде фіксувати натискання клавіш та зберігати їх у лог-файлі. Через певні проміжки часу даний лог-файл надсилається на сервер зловмисника, використовуючи поштовий агент. Кейлегери можуть створити такого поштового агента на цільовому пристрої. У деяких випадках поштовий агент використовує порти 80 або 587 для зв'язку. Зазвичай лог-файл, що створюється кейлогерами є схованим, іноді використовується шифрування. Розміру лог-файлу підтримується у певному діапазоні.[29]

Метод, що описаний далі базується на об'єднані обслідування пам'яті та моніторингу мережі. Як зазначалось раніше деякі вили кейлогерів використовують оперативну пам'ять для зберігання лог-файлу. Для пошуку процесу використовується аналіз інформації оперативної пам'яті та за можливості - аналіз мережевого трафіку. Нижче представлена послідовність дій для виявлення кейлогерів:

1) Запуск засобів для перехоплення мережевого трафіку (*tcpdump, Wireshark*)

2) Введення якогось тексту

3) Пошук SMTP пакету серед перехоплених пакетів

4) У випадку відсутності SMTP-пакету, можливо, кейлогер не використовує поштовий-агент, тому переходимо до наступного пункту

5) Робимо знімок пам'яті

6) Запускаємо Volatility framework для роботи з отриманим знімком

7) Отримуємо список процесів

8) Запускаємо сканування процесів на ознаки шкідливого програмного забезпечення

9) При відсутності результату з попереднього пункту спробувати знайти файл, що містить текст, що вводився раніше; Якщо знайдено файл, що містить введений текст, це може означати наявність кейлогера у системі

На рисунку 2.2 можна побачити алгоритм дій для вивлення кейлогерів за допомогою аналізу мережевого трафіку та оперативної пам'яті.

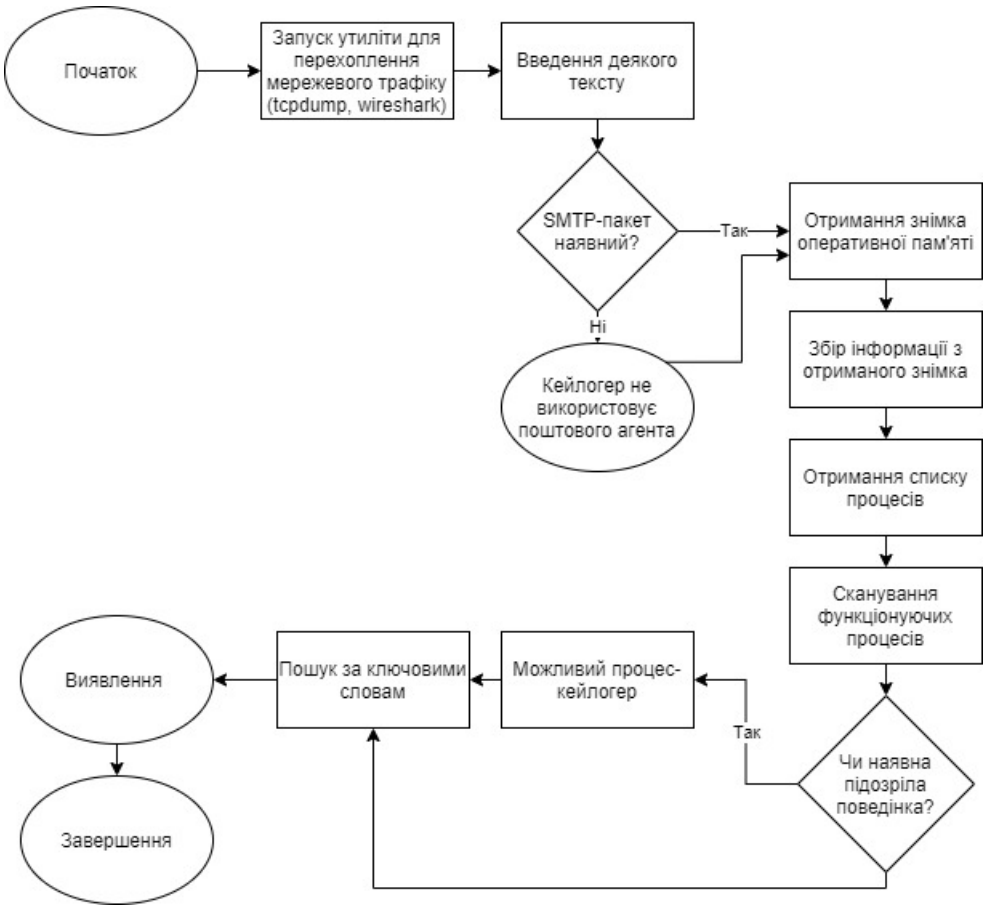

Рисунок 2.2 – Послідовність дій при виявленні кейлогерів за допомогою аналізу мережевого трафіку та оперативної пам'яті

Аналіз пам'яті

Експертизою пам'яті називають аналіз оперативної пам'яті. Це дозволяє знайти підозріле програмне забезпечення, що може бути шкідливим для пристрою. Через велику кількість технік приховування зловмисного програмного забезпечення експертиза пам'яті є однією з найбільш дієвих та ефективних способів їх виявлення. З допомогою експертизи пам'яті можна отримати велику кількість корисної інформації про функціонування системи:

– інформацію про функціонуючі процеси системи
– інформацію про виконуванні файли та її історію
– інформацію про користувача

Аналіз процесів функціонуючої системи є нетривіальною задачею. Тому для спрошення використовують дампи пам'яті, що являють собою точну копію оперативної пам'яті у певний момент часу.

2.3 Перешкодження роботі кейлогера за рахунок використання часових проміжків

Коли мова йде про шкідливе програмне забезпечення не завжди є можливість виявити його присутність. Тому можна використовувати методи, що мінімізують його впилив на систему. У випадку з кейлогерами користувач може не підозрювати про наявність такого програмного забезпечення на пристрої та вводити конфіденційні дані. Дана проблема стає ще більш актуальною, коли мова йде про використання не свого особистого пристрою.[4]

При використанні веб-браузера для доступу до сайтів та особистих

сторінок з'являється необхідність введення конфіденційної інформації, такої як пароль чи номери банківських карт. Для безпечного вводу можна застосувати підхід, при якому спеціальний застосунок генерує випадкові символи і в той же час для користувача виділяються невеликі проміжки часу впродовж яких він може вводити необхідну йому інформацію. Надалі даний метод буде описаний більш детально.

Метод на основі введення у виділенні проміжки часу дозволяє користувачу безпечно вводити конфіденційну інформацію при використанні веб-браузера. При даному методі натиснені користувачем клавіші записуються лише у проміжок часу, що визначається відповідним. Цей відповідний проміжок знаходиться між двома невалідними. [4] Після цього літери, що були введені у відповідні проміжку часу конкатенуються. Для того, щоб кейлогер не міг виявити, що у системі функціонує подібний застосунок легітимні проміжки часу, їх тривалість та розміщення відносно часової лінії повині змінюватись у деякому інтервалі. На рисунку 2.3 зображено, як може виглядати вікно для введення конфіденційної інформації у відповідні проміжки часу.

Рисунок 2.3 – Приклад вікна для введення інформації у задані проміжки часу

Для використання такого підходу можливим варіантом реалізації є використання його у формі доповнення для браузера, що буде

активуватись у момент, коли необхідно заповнювати поля з конфіденційною інформацією.

Хоча даний підхід і може бути ефективним, однак він має ряд своїх недоліків. Якщо атакуючий має змогу отримати декілька екземплярів з однією і тією ж конфіденційною інформацію, для прикладу паролем, він значно відвищить шанс його отримання.

Окрім цього важливо відмітити, що для використання такого методу існує необхідність попередньої практики, що може стати проблемою для широкого розповсюдження даної технології.

Висновки до розділу 2

У даному розділі було проведено огляд методів виявлення та мінімізації впливу кейлогерів. Було розглянуто метод виявлення кейлогерів у системі, що базується на аналізі мережевого трафіку та оперативної пам'яті для виявлення процесів, що можуть поводитись, як шкідливе програмне забезпечення типу кейлогер. Додатково було наведено метод, що дозволяє безпечно вводити конфіденційні дані при використанні веб-браузера. Цей метод дозволяє мінімізувати можливі втрати при використанні незнайомих пристроїв.

З вище зазначених тверджень було виявлено, що подібний підхід при виявленні кейлогерів є нетривіальною задачею, яка потребує спеціальних технічних знань, що може стане на заваді його розповсюдження. А метод, що базується на введені даних у відповідні інтервали часи потребує попередньої підготовки користувачів.

3 МЕТОД ВИЯВЛЕННЯ КЕЙЛОГЕРІВ, ЩО НЕ СПИРАЄТЬСЯ НА АРХІТЕКТУРУ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Підхід

Підхід, що описаний у даному розділі може бути ефективним при створенні засобів виявлення програмних кейлогерів простору користувача, що працюють у непривілейованому режимі. Даний тип кейлогерів являє собою фоновий процес, що фіксує натиснені користувачем клавіші, таким чином прослуховуючи можливу конфіденційну інформацію. Тобто метою даної роботи є створення моделі засобу для виявлення кейлогерів для унеможливлення викрадення конфіденційної інформації шляхом зчитування натиснених клавіш.

Модель базується на можливості помістити кейлогер у контрольоване середовище, де його поведінка буде безпосередньо фіксуватись та аналізуватись. Техніка, що використовується передбачає контроль даних на вході у кейлогер та подальший моніторинг його активності на виході. В незалежності від перетворень яким піддаються дані при проходжені через кейлогер можна вивести закономірність між вхідними та вихідним потоками. Окрім того, використовуючи спеціально згенеровані послідовності можна досягти більш точного виявлення паттернів поведінки кейлогерів.

Основною перевагою даного підходу є те, що він не залежить від внутрішньої структури кейлогера, від його архітектури. Окрім цього моніторинг вводу-виводу може виконуватись паралельно для декількох процесів. Таким чином даний підхід дозволяє перевірити всі процеси у системі для виявлення можливих кейлогерів у просторі користувача. Даний підхід ігнорує зміст вхідних та вихідних даних, фокусуючись лише на їх розподілі.

3.2 Архітектура

Архітектура застосунка складається з декількох модулей, що будуть виконувати певні дій на кожному з етапів аналізу. Модель застосунку можна побачити на рисунку 3.1 До цих компонентів відноситься інжектор, монітор, транслятор шаблонів, детектор та генератор шаблонів. Через те, що операційна система не надає деталі на більш високі рівні без привілейованих викликів API, інжектор та монітор функціонують на іншому рівні абстракції. На цьому рівні події натискання на клавіші та вивід байтів відбувається у вигляді потоку з певною швидкістю.

Завданням інжектора є введення потоку натискання клавіш для імітації поведінки користувача, котрий щось друкує на клавіатурі. Схожим чином монітор фіксує потік байтів на виході конкретного процесу. Інжектор отримує вхідний потік з транслятора шаблонів. Подібним чином монітор надсилає вихідний потік до транслятора шаблонів для подальшого аналізу.

Рисунок 3.1 – Архітектура застосунка

Інжектор

Задачею інжектора є введення вхідного потоку, що імітує поведінку користувача у системі. За конструкцією інжектор має задовольняти деякі принципи: 1) використання лише непривілейованих викликів API, 2) інжектор повинен мати змогу виконувати ін'єкції натискання клавіш зі змінною швидкістю, для відповідності розподілу вхідного потоку. Тобто кейлогер простору користувача не повинен відрізняти події викликані інжектором та користувачем. Для цього можна використати функціональність, що надається API викликом SendInput у сімействі систем Windows.

Монітор

Монітор виконує задачу запису вихідного потоку запущених процесів. Так же як і у випадку інжектора дозволяються лише непривілейовані API виклики. Для побудови вихідного потоку певного процесу, монітор надсилає запити на отримання даної частини інформації через регулярні проміжки часу та реєструє записану кількість байтів щоразу після кожного запиту. Для цього можна використовувати виклики програмних інтерфейсів, що надаються операційною системою, для доступу до структур, якими оперують процеси.

Транслятор шаблонів

Задачею транслятора шаблонів є перетворення абстрактного шаблону натискання клавіш у потік і навпаки, залежно від конфігураційних параметрів. Шаблон у формі АШНК[1] може бути змодельований як послідовність зразків, що виходять з потоку через рівномірні інтервали часу. Так зразок P_i шаблону P є абстрактним поданням кількості натискань клавіш протягом інтервалу часу i. Кожний зразок зберігається у нормалізованій формі в інтервалі **[0, 1]**, де 0 та 1 відображають заздалегідь визначений мінімум та максимум кількості натискань клавіш за певний час. Для перетворення шаблону у потік натискань клавіш, транслятор патернів розглядає наступні конфігураційні параметри: N - кількість входження зразка у шаблоні; T - постійний інтервал між двома послідовними зразками; K_{min} - мінімально дозволена кількість натискань клавіш для кожного зразка; K_{max} - максимальна кількість натискань клавіш, що дозволена для одного зразка.

При перетворенні введенного шаблону у формі АШНК у вхідний потік, транслятор патернів генерує для кожного часового інтервалу i потік натискань клавіш з деякою середньою швидкістю натискання $R_i = \frac{P_i(K_{max}-K_{min})+K_{min}}{T}$. Дана ітерація повторюється N разів, щоб покрити всі зразки в оригінальному шаблоні. Подібним чином виконується і перетворення потоку вихідних байтів у шаблон у формі АШНК. $P_i = \frac{R_i*T-K_{min}}{K_{max}-K_{min}}$. Де R_i - це середній темп натискання клавіші впродовж інтервалу i. Транслятор шаблонів повторно використовує ті ж самі параметри, що й у фазі генерації та знаходить P_i, знаючи попередньо виміряні значення середньої швидкості натискання клавіш.

[1] Абстрактний шаблон натискання клавіш

Детектор

Ефективність даного підходу полягає у здатності зробити висновки з причино-наслідкового зв'язку між потоком натискання клавіш, що ввів інжектор та поведінкою кейлогера на вході/виході його процесу. Хоча потрібно вивчати кожний процес у системі, алгоритм виявлення взаємодіє з одним процесом за раз, визначаючи, чи існує сильна подібність між вхідним шаблоном та вихідним, що отримані з аналізу поведінки цільового процесу.

Таким чином при побудові алгоритму виявлення першою задачею є прийняття відповідної метрики для вимірювання подібності двох заданих шаблонів. Використання абстрактних шаблонів натиснених клавіш дає змогу використати декілька можливих варіантів, щоб виконати порівняння двох шаблонів. Однак для початку можна зупинитьсь лише на одному зі способів порівняння і у подальшому для підвищення ефективності та точності можна використати декілька алгоритмів порівняння. Для визначення взаємозв'язку між послідовностями використовується міра кореляції. [20] Запропонований алгоритм виявлення базується на коефіцієнті кореляції Пірсона. Значення, що надаються даним методом симетричні та варіюються у межах від -1 до 1, де 0 означає відсутність кореляції, а 1 або -1 повну пряму або зворотну кореляцію. Таким чином цікавить лише позитивне значення кореляції.

Для прикладу маємо шаблони P та Q з N екземплярами, таким чином отримуємо, що $r = \frac{cov(P,Q)}{\sigma P \sigma Q} = \frac{\sum_{n=1}^{N}(P_i - P)(Q_i - Q)}{\sqrt{\sum_{n=1}^{N}(P_i - P)^2}\sqrt{\sum_{n=1}^{N}(Q_i - Q)^2}}$.

$cov(P, Q)$ - це коваріація, σp та σq - відхилення, P та Q значення отриманих зразків натискання.

Коефіцієнт кореляції Пірсона є широко використовуваним при визначенні зв'язків при різних розподілах. [21] Він був обраний через його математичні властивості. Даний підхід дозволяє виміряти лінійну

залежність між двома зразками при цьому ігноруючи нелінійний зв'язок. Для запропонованого методу лінійна залежність ефективно апроксимує зв'язок між шаблонами на вході та виході, що був згенерований кейлогером. Робиться припущення, що кейлогер не виконує дій для визначення розподілу натиснених клавіш впродовж введення інформації, а лише виконує дій на основі одиночних натискань. Коефіцієнт Пірсона є нечутливим для змін місцерозташування чи масштабу, а саме немає різниці, якщо кожний отриманий зразок P_i був модифікованим у деякий вигляд $a * P_i + b$, де a та b є деякими випадковими константами. У найкращому випадку шаблон на виході повинен бути точною копією шаблону на вході, якщо натискання клавіш були надіслані до процесу кейлогера, однак зазвичай кейлогер змінює оригінальну структуру шаблону. Подібні зміни можуть виконуватись декількома способами.

Дуже часто кейлогери можуть кодувати кожне з натискань у послідовність декількох байтів. Для прикладу деякий кейлогер може генерувати вихідний потік даних, що одним байтов описує один символ, в той же час інший кейлогер використовує для зберігання кодів клавіш 2 байти інформації. Таким чином шаблон згенерованого потоку буде мати однаку форму, однак у другому вона буде розтягнута, тобто змінена у масштабі. Однак, як зазначалось вище коефіцієнт кореляції, що використовується є стіким для подібних змін. Таким чином кейлогер може використовувати і більшу кількість байтів для кодування символів, що не повино впливати на кінцевий результат порівнянь, адже зміниться лише масштаб, а не форма.

Для функціонування кейлогери часто використовують деякий буфер фіксованого розміру, який заповнюється перехопленими клавішами та надсилається через фіксовані проміжки часу чи при повному його заповнені. Таким чином може виникнути ситуація, що буфер кейлогера є меншим за кількість клавіш K_{min}, що була натиснена. Тобто деяка частина згенерованої послідовності не потрапить до цього буферу та буде відсутня у вихідному потоці. У такому випадку на кожному інтервалі

ін'єкцій послідовностей даний буфер буду переповнюватись, та кількість байтів, що будуть у наступному буфері буде такою, що рівна кількості незаписаних байтів до попереднього буферу. Відносно цього можна ввести деяку зміну z, що буде відображати середню кількість натискань, що були відкинуті на кожному часовому відрізку. Даний параметр можна у подальшому врахувати та він не буде впливати на значення кореляції. Тому наступним важливим зауваженням є важливість коректного вибору розміру мінімальної кількості натискань клавіш K_{min}.

Важливим фактором для розгляду є вибір кількості зразків, що зберігаються. У даному питані важливо досягти балансу між точністю та швидкістю. Адже при невеликій кількості зразків для порівняння результати можуть мати недостатню точність виявлення. В той же час при підвищенні кількості цих зразків точність результатів буде зростати та приведе до зменшення впливу можливих помилок вимірювання, однак призведе зо значного підвищення кількості часу необхідного для проведення аналізу. Так як алгоритм повністю покладається на механізм кореляції, то необхідно запровадити деякі порогові значення для виявлення можливих шкідливих процесів. Однак необхідно зазначити, що при даному підході можливі хибно позитивні результати.

Генератор шаблонів

Генератор патернів має підтримувати декілька алгоритмів генерації шаблонів. Важливим аспектом при генерації шаблонів є створення вибірок, що розподілені у більш широкому діапазоні, так як це дає більш сильні кореляції. Тобто чим більша мінливість у даних розподілах, тим стабільнішим і точнішим є обчислення коефіцієнта кореляції Пірсона. Окрім цього чим ширше діапазон значень, що обмежуються

максимальним та мінімальними значеннями швидкості натискання клавіш, тим надійнішими є результати.

При розробці генератора шаблонів важливо звернути увагу на можливість генерування більш випадкових та різноманітних вхідних шаблонів. Велика кількість робіт з аналізу ефективності кореляції Пірсона показує важливість вибору більше зразків у більш широкому діапазоні значень для більш сильної кореляції. Тобто більша різноманітність у розподілі дає більш стабільні та точні результати при розрахунках коефіцієнта кореляції. Таким чином вхідний шаблон має містити зразки з усього інтервалу **[0,1]**. Степінь варіативності результуючому вхідному потоці також залежить від діапазону швидкості натискання клавіш у трансляторі шаблонів. Чим ширшим є діапазон між мінімальною та максимальною швидкістю, тим точнішими є результати.

Негативні наслідки невеликої варіативності у вхідному шаблоні можна вивести із математичних властивостей даної кореляції. Чим ближче патерни розподілені до крайніх значень, тим менш точним є коефіцієнт кореляції. У випадку, коли відхилення рівне 0 коефіцієнт кореляції визначити неможливо. Звідси можна зробити висновок, що найдійний генератор патернів не повинен розглядати константні або маловаріативні патерни.

Ефективний алгоритм для генерації патернів в ідеалі повинен давати невелику кількість хибно-позитивних результатів. У випадку, коли з вихідного потоку генерується константний патерн, то йому призначається рівень кореляції рівний 0. Так як вхідний потік був згенерований з задовільним рівнем варіативності.

Для підведення підсумків щодо підходів при розробці алгоритмів генерації шаблонів необхідно зазначити наступні методи:

– Випадковий - кожний зразок генерується у випадковий спосіб

– Патерн генерується за рахунок перестановок деякої кількості зразків, що є рівномірно розподіленими на відрізку

3.3 Можливі методи приховання та атаки спрямовані на детектор

Збільшення розміру буфера

Для протидії виявлення при розробці кейлогерів можуть використовувати більш великі об'єми буферів для зберігання даних. При значному збільшенні розмірів інтервалів через які відбувається надсилання інформації та розмірів самого буфера, запропонована модель має ряд недоліків. У випадку надто великого інтервалу збору даних такий підхід є непрактичним, через необхідність великої кількості часу для аналізу та зберіганні великої кількості зразків у пам'яті пристрою, що у разі перевірки великої кількості працюючих процесів є неефективним методов чи навіть неможлиивим.

Однак подібне зауваження доцільне відносно розробленої техніки, а не моделі. Адже замість використання вхідних та вихідних потоків та знаходження кореляції між ними, можна використовувати подібний підхід при розробці шаблонів при доступі до файлової системи. Однак дана модифікація приведе до втрати такої переваги як використання непривілейованих функцій операційної системи.

Адаптація поведінки

Одним із можливих підходів для маскування своєї діяльності може стати адаптація поведінки кейлогера. Тобто даний кейлогер може бути розроблений, щоб перехоплювати натискання клавіш з якогось конкретного застосунку. Тому кейлогер може активізуватись у випадку

виникнення деякої події. Такою подією моєе бути запуск певного застосунку чи процесу.

Такий тип кейлогерів є невидимим для розробленої вище техніки. Однак подібна проблема присутня для більшості засобів виявлення шкідливого програмного забезпечення, що базуються на динамічному аналізі.

Виходом у даній ситуації може бути повторний запуск застосунку для виявлення кейлогерів, якщо користувач має на те підозри. І такі періодичні запуски можуть зменшити вірогідність того, що кейлогер буде продовжувати функціонувати.

Маскування при виявленні детектора

Спроби імітувати поведінку користувача можуть стати причиною виявлення функціонуючого детектора кейлогером, що здатен відрізняти реальну поведінку користувача та штучно згенеровану. Використовуючи статистичні характеристики мови, такі як частотний аналіз, що показує частоту тої чи іншої букви у словах певної мови, кейлогер може ігнорувати потоки натискань клавіш, що генерує детектор.

Однак подібна проблема доволі просто розв'язується, якщо дещо ускладнити генератор патернів та модифікувати його для того, щоб він генерував статистично коректні послідовності клавіш. Окрім цього при надсиланні згенерованих натискань важливо підтримуватись темпу, що є можливим при справжньому наборі людини, інакше через дуже великий темп кейлогер може ігнорувати введені послідовності. Хоча такий підхід може стати причиною відмови роботи кейлогера, так як постійні ввімкнення та вимкнення його функціонування може спровокувати відмову в обслуговуванні, адже сама система генерує велику кількість

хибних повідомлень.

Атака спрямована на кореляцію

При атаках на кореляцію відбуваються спроби зменшити кореляцію між вхідним та вихідним потоками. Одним з можливих сценаріїв проведення подібної атаки може бути генерація власного шаблону кейлогером та створення шуму на вихідному потоці з ціллю зменшити кореляцію. Однак при такому підході шкідливому застосунку буде необхідно при кожному новому патерні на вході генерувати власний патерн з іншим розподілом на виході.

Для протидії цій атаці можна використати наступний підхід. Процеси позначаються як нешкідливі у разі, коли при зміні алгоритму генерації вхідного шаблону поведінка вихідного потоку процесу, що аналізується, не змінюється.

В той же час для виявлення нелегітимних процесів можна використати дещо інший алгоритм. На початку генеруються не випадкові шаблони для кожного з функціонуючих процесів у системі. Дані шаблони зберігаються для подальшого порівняння. Після цього відбувається генерація шаблонів за алгоритмом з випадковими вхідними символами. Після цього порівнюються вихідні потоки процесів, що були згенеровані при першому кроці з тими, що були отримані на другому. Наприкінці процес позначається як нелегітимний, якщо обчислена подібність не перевищує деяке порогове значення.

3.4 Рекомендації щодо програмної реалізації

У даній роботі розглядаються кейлогери простору користувача. Даний вид кейлогерів був обраний через те, що вони є найбільш популярними серед програм для нелегітимного отримання конфіденційної інформації. Їх популярність викликана простотою реалізації та відсутності необхідності у привілеях адміністратора. Надалі буде надано вказівки та опис підходів та функцій, які рекомендовано використовувати при розробці застосунку для виявлення програмних кейлогерів простору користувача. Мова піде про систему Windows, як про найбільш розповсюджену серед звичайних користувачів.

Архітектура застосунка

Для початку необхідно окреслити модель, за якою буде розроблятись програмний застосунок. У попередній розділах при описі моделі та алгоритму виявлення було зазначено, що програмна реалізація буде складатись з 5-ти модулей з різною функціональністю. На рисунку 3.2 зображено схему застосунка.

Як видно з рисунку у головній функції для доступу до інформації про функціонуючі процеси системи використано *Windows Snapshot Tool*. Дана функція дозволяє зробити знімок системи, а саме стан процесів, потоків та модулів, що функціонують та використовуються даними процесами. Для взаємодії із процесами та їх потоками необхідно знати їх ідентифікатори. Дізнатись це можна звертаючись до структур Windows, що використовують для збереження інформації про процеси. На рисунку 3.3 зображено приклад використання засобів Windows для отримання

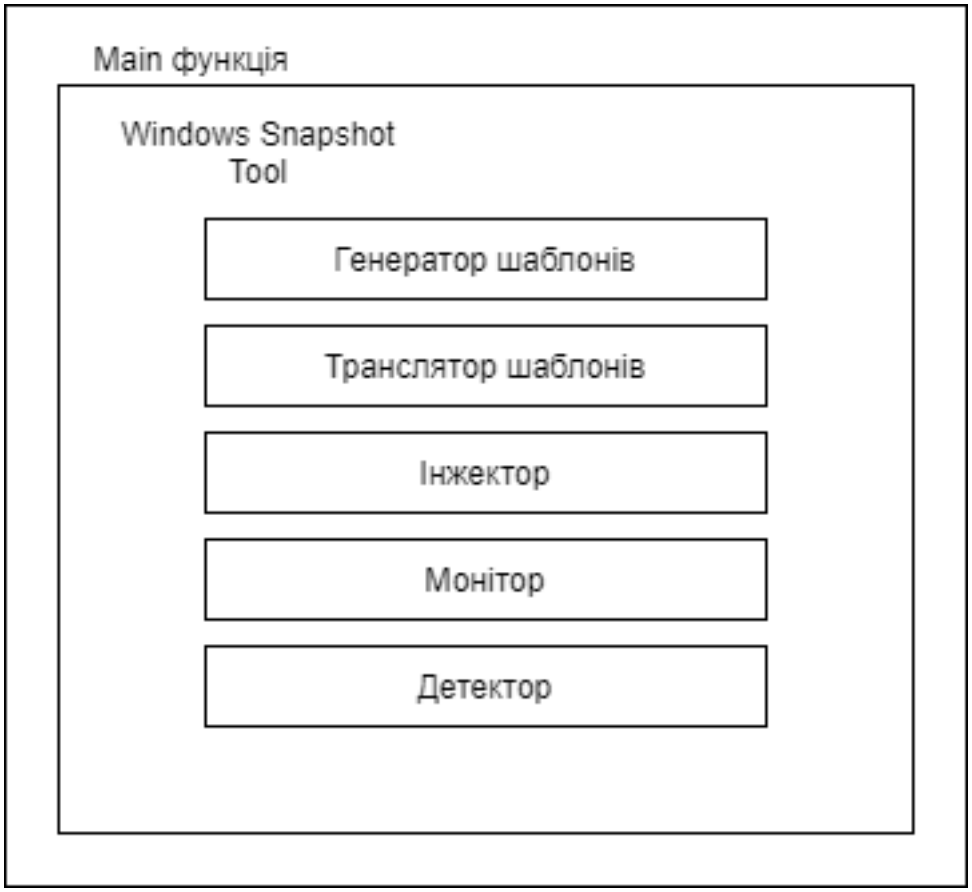

Рисунок 3.2 – Схема застосунка

списку процесів, що наразі функціонують у системі та інформацію про їх потоки та пріоритет.

В той же час окремі функціонуючі модулі можуть бути реалізовані, або як окремі виконувані файли, тобто окремі застосунки, чи у вигляді функцій. У нашому випадку було обрано другий варіант. Генератор шаблонів, що є початковою ланкою у алгоритмі, відповідає за створення патернів, які надалі будуть трансформуватись у спеціальний вигляд. Як описувалось у розділі про генератор шаблонів, він має підтримувати декілька режимів при генерації патернів.

Одним з цих режимів є режим створення патерну випадковим чином. Операційна система надає можливість для генерації випадкових чисел. Хоча для криптографічних цілей подібні генератори не використовуються через їх невелику періодичність та таким чином можливість вирахування наступного згенерованого значення, для наших цілей вбудованих

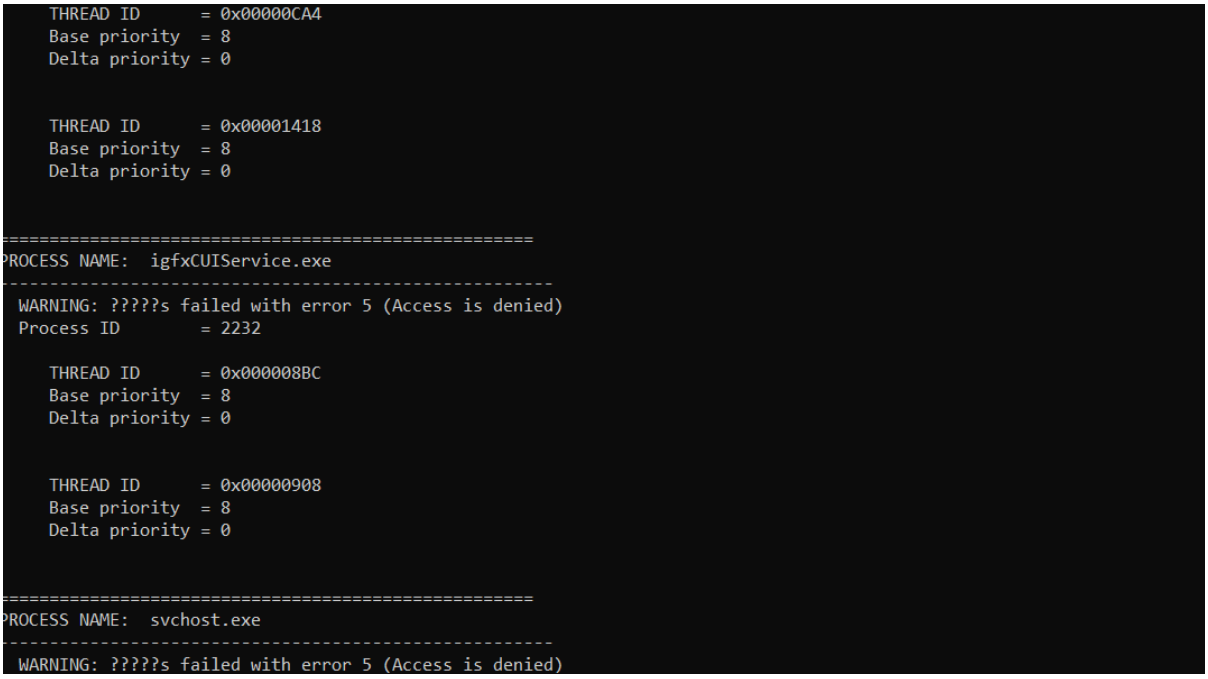

Рисунок 3.3 – Список процесів, що запущені у системі

можливостей буде достатньо.

Символи у операційній системі описуються віртуальними кодами клавіш. Для цього використовується ASCII, у даній таблиці, у розширеній її версії присутні 256 значень за кожним з яких закріплений конкретний символ.

Так як кейлогери перехоплюють натиснені користувачем клавіші, то нам необхідно випадковим чином генерувати символи. Для цього можна використати наступний підхід:

1) Запуск циклу з генератором всередині

2) Генерація числа при кожному такті циклу з використанням поточного часу як зерна для генерації різних випадкових значень

3) Від отриманого числа береться модуль 255 та залишок і буде кодом клавіші

Після генератора шаблонів отримана послідовність направляється до транслятора шаблонів. Для передачі інформації між функціями найбільш простим варіантом буде використання посилань на структури з описаними послідовностями у тілі функції. На кожному такті дані структури будуть перезаписуватись новими згенерованими значеннями.

Задачею транслятора шаблонів є перетворення абстрактних шаблонів натискання клавіш у потік байтів і навпаки. На попередньому етапі при генерації шаблону отримується потік байтів, що описують згенеровані послідовності. Транслятор переводить їх у вигляді при якому для кожного згенерованого зразка підраховується значення P, що відображає кількість натискань певної клавіші впродовж фіксованого інтервалу часу.

Наступною ланкою роботи алгоритму є інжектор, котрий буде надсилати згенеровані послідовності у процеси, що функціонують у системі. Для цього необхідно використовувути API-функції, що надаються операційною системою. У версія Windows до Windows 7 для цього можна було використовувати функцію *keybd_event*, вона генерувала повідомлення *WM_KEYUP* та *WM_KEYDOWN*, що імітували натискання клавіші. Однак у сучасних версіях Windows дана функція була замінена на SendInput. Проблемою останньої є те, що вона надсилає натискання клавіш лише до процесів з активними вікнами. Так як мова йде про роботу з усіма процесами у системі, то більшість з них не має вікон, тому для імітації натискань можна використати функцію *PostMessage* чи *PostThreadMessage*. Остання приймає чотири параметри: ідентифікатор потока процеса; повідомлення, що у нашому випадку є *WM_KEYUP* та *WM_KEYDOWN*; віртуальний код натисненої клавіші.

Інжектор повинен мати можливість проводити відправлення повідомлень з певною періодичністю та темпом. Для цього знову можна використати системну функцію *Sleep()*, що призупиняє виконання активного потоку на зазначений інтервал часу. А даний інтервал отримується за рахунок використання генератора випадкових чисел для отримання значення затримки у деякому діапазоні значень.

Для збірки та подальшого аналізу потоку використовується модуль монітор. З цією метою можна використати лічильники продуктивності для процесів, що надаються операційною системою. У сімействі систем Windows з цією метою може бути використаний такий інструмент як *WMI(Windows Management Instrumentation)*. У системі Windows для

опису та взаємодії із процесами використовується класс *Win32_Process*. У ньому наявна зміна, що описує кількість байтів, що була записана з моменту створення процесу. Таким чином через фіксовані інтервали часу можна записувати кількість записаних байтів впродовж даного інтервалу, щоб у подальшому з цього зібрати вихідний потік.

Описаний вище підхід при програмній реалізації базується на розрахунках лише у одному потоці. Це може стати причиною того, що для аналізу процесів буде необхідно велика кількість часу. Тому для більш ефективного функціонування необхідно переписати даний застосунок під виконання у декілька потоків одночасно, одна при цьому значно зросте складність програми та необхідна кількість ресурсів для функціонування. Адже при 4-х ядерному процесорі з 8-ю потоками виникне необхідність у одночасному функціонуванні 8-ми екземплярів кожного з модулей, що з ростом складності та варіативності коду може стати проблемою для малопотужних пристроїв.

Не менш важливим питанням при розробці багатопотокового застосунку буде синхронизація потоків. А саме задача у том, щоб зроботи так, щоб окремі екземпляри не обробляли один і той самий потік, адже у даному випадку будуть отримані некоректні дані.

Висновки до розділу 3

У даному розділі було представлено та описано метод для виявлення кейлогерів простору користувача. Він базується на відповідності між патернами, що генерує застосунок. Таким чином подібний метод не спирається на внутрішній устрій шкідливого програмного забезпечення типу кейлогера, а лише на його взаємодію із введеними даними. Додатково було наведено рекомендації щодо використання методів операційної системи при розробці застосунку.

ВИСНОВКИ

У ході роботи було проведено огляд та аналіз опублікованих джерел за тематикою виявлення шкідливого програмного забезпечення, методи виявлення кейлогерів простору користувача, принципи роботи та побудови кейлогерів, їх види та архітектура. Було наведено класифікацію кейлогерів в залежності від їх фізичних характеристик та види кейлогерів, що представляють собою програмні застосунки. У ході ознайомлення з існуючою літературою було проведено огляд сучасних методів виявлення кейлогерів на прикладі використання аналізу мережевого трафіку та оперативної пам'яті. Наведено метод, що дозволяє безпечно вводити конфіденційну інформацію у веб-браузері, що базується на використанні методу часових інтервалів. Для запропонованих методів виявлення та протидії було визначено слабкі сторони та складності, з якими може зіштовхнутись користувач.

У третьому розділі було наведено підхід для виявлення кейлогерів простору користувача, що базується на порівнянні спеціально згенерованих патернів. Даний метод не спирається на внутрішній устрій шкідливої програми типу кейлогер, його архітектуру. Додатково були надані рекомендації щодо використанння методів операційної системи для розробки застосунку.

Однак запропонований метод має деякі слабкі сторони, якими може скористатись зловмисник. Тому напрямом подальших досліджень може стати мінімізація можливості проведення успішної атаки на застосунок вивлення кейлогерів. Окрім цього важливим питанням є швидкодія застосунку, адже у рекомендаціях щодо розробки зазначено використання лише одного потоку при аналізі, що може стати причиною використання великої кількості часу для проведення аналізу. Тому при розробці краще спиратись на багатопоточну реалізацію, не дивлячись на складність синхронизації потоків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Security Technology Ltd. Testing and reviews of keyloggers, monitoring products and spyware— http://www.keylogger.org.

2. Olzak T. Keystroke Logging (Keylogging)— 2008. https://www.researchgate.net/publication/228797653_Keystroke_logging_keylogging

3. Creutzburg R. The strange world of keyloggers - an overview, Part I— 2017.— https://doi.org/10.2352/ISSN.2470-1173.2017.6.MOBMU-313.

4. Chien-Wei Hung, Fu-hau Hsu, Shih-Jen Chen, Chang-Kuo Tso, Yan-Ling Hwang, Po-Ching Lin, Li-Pin Hsu A QTE-based Solution to Keylogger Attacks— 2012.

5. Yahye Abukar, Mohd Aizaini Maarof, Fuad Mire Hassan Survey of Keylogger Technologies— 2014.— https://www.researchgate.net/publication/309230926_Survey_of_Keylogger_Technologies.

6. Moser A., Kruegel C, Kirda E. Exploring multiple execution paths for malware analysis— 2007.— Proc. of the 28th IEEE Symposium on Security and Privacy

7. Ortolani S., Giuffrida C., Crispo B. Unprivileged Black-box Detection of User-space Keyloggers— 2013.

8. Kruegel C, Kirda E. Behavior-based Spyware Detection— Secure Systems Lab

9. Saiganesan N., Dheenadhayalan A., Arulmani M., Suresh K. Anti-Hacking Mechanism for Keylogger using Blackbox Detection — ISSN:2278-0181 — International Journal of Engineering Research & Technology (IJERT)

10. Khaleel A., Doja M.N A Novel Framework for Password Securing System from Key-logger Spyware — Secure Systems Lab

11. A Malware Variant Detection Method Based on Byte Randomness Test — JOURNAL OF COMPUTERS,— VOL. 8, NO. 10, — OCTOBER 2013

12. Wajahat A., Imran A., Nazir A., Latif J. A Novel Approach of Unprivileged Keylogger Detection— International Conference on Computing,

Mathematics and Engineering Technologies— 2019

13. Muhammad A., Rana N.I., Mirza M.B., Arshad M.A. Anti-Hook Shield against the Software Key Loggers— Al-Khawarizmi Institute of Computer Science, University of Engineering and Technology— National Conference on Emerging Technologies— 2004

14. Jun Fu., Huan Yang Enhancing Keylogger Detection Performance of the Dendritic Cell Algorithm by an Enticement Strategy— JOURNAL OF COMPUTERS— VOL. 9, NO. 6,— JUNE 2014—

15. Heejo L. HoneyID : Unveiling Hidden Spywares by Generating Bogus Events— 2008

16. Hung C.W., Hsu F.H., Wang C.H., Lee C.H. Keyloggers Prevention with Time-Sensitive Obfuscation— World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering — Vol:14 — No:6 — 2020

17. Heejo L. HoneyID : Unveiling Hidden Spywares by Generating Bogus Events— 2008

18. Huseynov H., Saadawi T., Igbe O., Kourai K. Virtual Machine Introspection for Anomaly-Based Keylogger Detection— 2020

19. Vuagnoux M., Pasini S..Compromising electromagnetic emanations of wired and wireless keyboards— 2009

20. Rodgers J., Nicewander W.Thirteen ways to look at the correlation coefficient— The American Statistician — pp. 59–66 — 1988

21. Benesty J., Chen J., Huang Y. On the importance of the pearson correlation coefficient in noise reduction— IEEE Trans. on Audio, Speech, and Language Processing— 2008

22. Xu M., Salami B., Obimbo C. How to protect personalinformation against keyloggers— Proc. of the 9th Intl. Conf. on Internet and Multimedia Systems and Applications— 2005

23. Kelly A., Chen J., Huang Y. Cracking Passwords using Keyboard Acoustics and Language Modeling,— 2010

24. Barisani A., Bianco D.Sniff Keystrokes With Lasers/Voltmeters - Side

Channel Attacks Using Optical Sampling Of Mechanical Energy And Power Line Leakage,— DEFCON 17— 2009

25. Case A., Golden G.Memory forensics: The path forward.— Digital Investigation 20— 2017

26. Kolte M.Unprivileged detection of user space keyliggers.— MITCOE International Journal of Innovation Research in Science

27. Le D., Yue C., Smart T., Wang H.Detecting Kernel Level Keyloggers Through Dynamic Taint Analysis,— Technical Report, College of William Mary— pp 3–5— 2008

28. Kruegel C., Valeur F., Robertson W., Vigna G.Static Analysis of Obfuscated Binaries— 2004

29. Ahmed B., Shoikot M., Hossain J., Rahman A.Keylogger Detection using Memory Forensic and Network Monitoring— International Journal of Computer Applications — Volume 177 – No. 11 — 2019

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Приклад використання функції *PostThreadMessage*:

```
HWND Inject(DWORD dwProcessId)
{
    HWND hwnd = NULL;
    do {
    hwnd = FindWindowEx(NULL, hwnd, NULL,
    (LPCWSTR)"notepad.exe");
    DWORD dwPID = 0;
    DWORD hThread = GetWindowThreadProcessId(hwnd, &dwPID);
    if (dwPID == dwProcessId && hThread != NULL) {
        PostThreadMessage(hThread, WM_KEYDOWN, 0x41, 1);
        PostThreadMessage(hThread, WM_KEYUP, 0x41, 1);
    }
    else {
        cout << GetLastError() << endl;
        }
} while (hwnd != 0);
return hwnd;
}
```