УДК 004.492.3

СПОСОБИ ВИЯВЛЕННЯ КЕЙЛОГЕРІВ У ПРОСТОРІ КОРИСТУВАЧА

М. А. Топчій[1, a], Л. Ю. Гальчинський[1]

[1] *Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Фізико-технічний інститут*

Анотація

Програмні кейлогери є одним з найпоширеніших способів, що використовується для збору конфіденційної інформації. Причиною на те є можливість працювати програмам у просторі користувача при записі послідовності клавіш, що натискає користувач. Можливість працювати у просторі користувача дозволяє більш просто розробляти подібне програмне забезпечення та розповсюджувати його, але у той же час дозволяє зрозуміти модель поведінки таких засобів. Відповідно до цього можна розробити методику, використовуючи спеціально згенеровані послідовності символів на вході та слідкуючи за поведінкою процесів на виході визначати такі програмні засоби серед запущених процесів

Ключові слова: Програмне забезпечення, кейлогери, безпека

Вступ

Кейлогери встановлюються на пристрої для моніторингу за діяльністю користувача шляхом фіксації клавіш, що натискаються, та подальшому відправленні даних послідовностей на сервери зловмисників. Зазвичай кейлогери використовуються для викрадення конфіденційної інформації, так викрадається велика кількість паролів, номерів кредитних карт,що ставить даний вид шкідливого програмного забезпечення на місце одних з найбільш небезпечних шпигунських програм[3][4].

Кейлогери можуть бути реалізовані у вигляді невеликих закладних апаратних пристроїв чи у вигляді програмного забезпечення. Кейлогери у вигляді програмного забезпечення можна додатково класифікувати в залежності від привілеїв, що необхідні для функціонування. Кейлогери, що написані на рівні ядра працюють з повними привілеями. В той же час можлива реалізація у непривілейованому режимі, де кейлогер запускається як звичайний процес простору користувача. Зазвичай кейлогери рівня користувача використовують набори непривілейованих API[1] доступних на сучасних операціних системах. Даний підхід значно спрощує розробку програмного забезпечення, адже розробка кейлогера на рівні ядра вимагає значно більших зусиль та знань з впровадження та відлагодження.[1][9]

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

1. Принципи роботи сучасних кейлогерів

Отримання конфіденційної інформації особи, реєструючи натискання клавіш може бути здійснене на багатьох різних рівнях. Прикладом є використання апаратного кейлогера у випадку, коли зловмисник має фізичний доступ до пристрою. У даному випадку кейлогер представляє собою засіб, що вмонтовується між материнською платою та роз'єму клавіатури. Відомі також варіанти використання таких фізичних явищ, як акустичні випромінювання, що створюються при наборі тексту; електромагнітне випромінення плати для бездротових пристроїв.

Через необхідність фізичного доступу до пристроїв при використанні апаратних кейлогерів більш поширеними є програмні кейлогери. Серед них виділяють кейлогери на основі гіпервізора, що являє собою шар між апаратним забезпеченням та операційною системою. Кейлогери ядра зазвичай реалізуються як частина більш складного руткіта, відмінність полягає у тому, що хуки використовуються беспосередньо для перехоплення подій обробки буферу чи інших повідомлень ядра.

Кейлогери рівня ядра ефективні, але їх використання вимагає привілейований доступ до пристрою. В той же час кейлогери простору користувача не потребують спеціальних прав для розгортання у системі. Їх виконання не залежить від наданих привілеїв. Окрім цього кейлогери простору користувача значно простіші у розробці через добре описані та відлагоджені API[1].

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

[a] nikita.topchiy@gmail.com

[1] Application programming interface – прикладний програмний інтерфейс

рів можуть бути реалізовані засобами Windows без використання додаткових бібліотек.

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

2. Підхід

Підхід, що описаний у даній статті може бути ефективним при створенні засобів виявлення програмних кейлогерів простору користувача, що працюють у непривілейованому режимі.[1] Даний тип кейлогерів являє собою фоновий процес, що фіксує натиснені користувачем клавіші, таким чином прослуховуючи можливу конфіденційну інформацію. Тобто метою даної роботи є створення моделі засобу для виявлення кейлогерів для унеможливлення викрадення конфіденційної інформації шляхом зчитування натиснених клавіш.

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

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

3. Архітектура

Архітектура застосунка складається з декількох модулей, що будуть виконувати певні дій на кожному з етапів аналізу. Модель застосунку можна побачити на рисунку 1 До цих компонентів відноситься інжектор, монітор, транслятор шаблонів, детектор та генератор шаблонів. Через те, що операційна система не надає деталі на більш високі рівні без привілейованих викликів API, інжектор та монітор функціонують на іншому рівні абстракції. На цьому рівні події натискання на клавіші та вивід байтів відбувається у вигляді потоку з певною швидкістю.

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

Рис. 1. Архітектура застосунка

3.1. Інжектор

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

3.2. Монітор

Монітор виконує задачу запису вихідного потоку запущених процесів. Так же як і у випадку інжектора дозволяються лише непривілейовані API виклики. Для побудови вихідного потоку певного процесу, монітор надсилає запити на отримання данної частини інформації через регулярні проміжки часу та реєструє записану кількість байтів щоразу після кожного запиту.

3.3. Транслятор шаблонів

Задачею транслятора шаблонів є перетворення абстрактного шаблону натискання клавіш у потік і навпаки, залежно від конфігураційних параметрів. Шаблон у формі АШНК[2] може бути змодельований

[2]Абстрактний шаблон натискання клавіш

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

При перетворенні введенного шаблону у формі АШНК у вхідний потік, транслятор патернів генерує для кожного часового інтервалу i потік натискань клавіш з деякою середньою швидкістю натискання. Дана ітерація повторюється N разів, щоб покрити всі зразки в оригінальному шаблоні. Подібним чином виконується і перетворення потоку вихідних байтів у шаблон у формі АШНК. Транслятор шаблонів повторно використовує ті ж самі параметри, що й у фазі генерації та знаходить P_i, знаючи попередньо виміряні значення середньої швидкості натискання клавіш.

3.4. Детектор

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

Таким чином при побудові алгоритму виявлення першою задачею є прийняття відповідної метрики для вимірювання подібності двох заданих шаблонів. Для визначення взаємозв'язку між послідовностями використовується міра кореляції. Запропонований алгоритм виявлення базується на коефіцієнті кореляції Пірсона. Значення, що надаються даним методом симетричні та варіюються у межах від -1 до 1, де 0 означає відсутність кореляції, а 1 або -1 повну пряму або зворотну кореляцію. Таким чином цікавить лише позитивне значення кореляції.

3.5. Генератор шаблонів

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

Висновки

У даній роботі було представлено підхід для виявлення найбільш поширених кейлогерів таких, що працюють у просторі користувача, як звичайні застосунки, що запущені у непривілейованому режимі. Модель поведінки кейлогера була змодельована шляхом кореляції між потоком введення та потоком виведення. Крім того в доповнення було використано метод за якого вхідний потік був штучно згенерований для більшої точності виявлення.

Перелік використаних джерел

1. Ortolani S., Giuffrida C., Crispo B. Unprivileged Black-box Detection of User-space Keyloggers— 2013.
2. Security Technology Ltd. Testing and reviews of keyloggers, monitoring products and spyware— `http://www.keylogger.org`.
3. Olzak T. Keystroke Logging (Keylogging)— 2008. `https://www.researchgate.net/publication/228797653_Keystroke_logging_keylogging`
4. Creutzburg R. The strange world of keyloggers - an overview, Part I— 2017.— `https://doi.org/10.2352/ISSN.2470-1173.2017.6.MOBMU-313`.
5. Chien-Wei Hung, Fu-hau Hsu, Shih-Jen Chen, Chang-Kuo Tso, Yan-Ling Hwang, Po-Ching Lin, Li-Pin Hsu A QTE-based Solution to Keylogger Attacks— 2012.
6. Yahye Abukar, Mohd Aizaini Maarof, Fuad Mire Hassan Survey of Keylogger Technologies— 2014.— `https://www.researchgate.net/publication/309230926_Survey_of_Keylogger_Technologies`.
7. Moser A., Kruegel C, Kirda E. Exploring multiple execution paths for malware analysis— 2007.— Proc. of the 28th IEEE Symposium on Security and Privacy
8. Kruegel C, Kirda E. Behavior-based Spyware Detection— Secure Systems Lab
9. Saiganesan N., Dheenadhayalan A., Arulmani M., Suresh K. Anti-Hacking Mechanism for Keylogger using Blackbox Detection — ISSN:2278-0181 — International Journal of Engineering Research & Technology (IJERT)