สร้างโปรแกรม Backup และ Restore Router ผ่าน TFTP server ในโครงการ GIN จำนวน 646ตัว

ในระบบเครือข่ายองค์กรขนาดใหญ่ การสำรอง (backup) และกู้คืน (restore) ค่าการตั้งค่าของอุปกรณ์เครือข่าย (เช่น Switch หรือ Router) เป็นสิ่งสำคัญอย่างยิ่ง การทำขั้นตอนเหล่านี้ด้วยมือในแต่ละครั้งอาจก่อให้เกิดความล่าช้าและความผิดพลาดได้

เราจึงออกแบบและพัฒนาเครื่องมือ GUI ด้วยภาษา Python และรู้จักการสร้าง GUI ที่สามารถเชื่อมต่อกับอุปกรณ์ผ่าน Telnet และ SSH เพื่อดำเนินการสำรองและกู้คืนค่าการตั้งค่าแบบอัตโนมัติ โดยโครงการนี้ การ Backup ต้องใช้ Telnet บน Router แล้วสามารถ SSH ไปหา Router ที่ต้องการได้ ส่วนการ Restore ต้องใช้การ Telnet ไปที่ ตัวRouter นั้นๆเลย

1. ฟีเจอร์หลัก

1.1.Backup แบบอัตโนมัติ: ใช้ Telnet เข้า jump host , SSH ต่อไปยังอุปกรณ์ ,สั่ง backup ด้วยคำสั่ง copy running-config tftp:

1.2 Restore แบบกู้ค่าการตั้งค่า:เชื่อมต่อ Telnet ไปยังอุปกรณ์ , ใช้คำสั่ง copy tftp: running-config

1.3 รองรับการทำงานหลายอุปกรณ์พร้อมกัน (multi thread) เข้าด้วย 10 Session

1.4เช็คสถานะ online / offline ก่อนดำเนินการด้วย ping

1.5 แสดง log และสรุปผลแบบ real-time

1.6 บันทึกผลลัพธ์เป็นไฟล์ CSV พร้อมสรุปจำนวนสำเร็จ/ล้มเหลว/skip

2.Coding

2.1 Library

import threadingใช้สำหรับ รันฟังก์ชันต่าง ๆ แบบเบื้องหลัง (Background Thread) เช่น การสำรองข้อมูลหลาย IP พร้อมกัน โดยไม่ให้ GUI ค้าง

import tkinter as tk tkinter คือไลบรารีมาตรฐานของ Python สำหรับสร้าง GUI (ปุ่ม ช่องกรอก ข้อความ)

from tkinter import scrolledtext, ttk, filedialogนำเข้าส่วนเสริมของ tkinter: เช่น scrolledtext: กล่องข้อความที่มี scroll bar ในตัว (สำหรับ log) , ttk: ธีมใหม่ของ tkinter (ใช้ใน Treeview และปุ่มต่าง ๆ) , filedialog: ใช้เปิด dialog ให้ผู้ใช้เลือกไฟล์ .txt หรือ .cfg

import telnetlib ไลบรารีสำหรับ เชื่อมต่ออุปกรณ์ผ่าน Telnet ,ใช้ในการ login เข้า switch/router และส่งคำสั่ง CLI

import timeใช้ฟังก์ชัน time.sleep() และจับเวลาผ่าน time.time() เพื่อคำนวณเวลาที่ใช้ backup แต่ละรอบ

import csv ใช้สำหรับสร้างและเขียนไฟล์ .csv เพื่อบันทึกผลลัพธ์ของการสำรองข้อมูลแต่ละ IP ลงในรูปแบบตาราง (Summary)

import os จัดการกับระบบไฟล์ในคอมพิวเตอร์

import platformใช้ตรวจสอบว่า OS ที่รันอยู่คือ Windows หรือ Linux/Mac เพื่อใช้ ping ให้ถูกต้อง

import subprocessใช้รันคำสั่ง ping ไปยัง IP ต่าง ๆ แล้วอ่านผลลัพธ์กลับมาเพื่อตรวจสอบว่าอุปกรณ์ ออนไลน์ หรือไม่

import concurrent.futures สำหรับ รันหลายฟังก์ชันพร้อมกัน (Multithread) อย่างปลอดภัย

import webbrowser ใช้เปิดโฟลเดอร์ที่เก็บไฟล์ summary ผ่าน default file explorer (เช่น File Explorer ใน Windows) โดยอัตโนมัติ

from datetime import datetime ใช้ดึงเวลาปัจจุบันมาใช้สร้างชื่อไฟล์

```
1   import threading
2   import tkinter as tk
3   from tkinter import scrolledtext, ttk, filedialog
4   import telnetlib
5   import time
6   import csv
7   import os
8   import platform
9   import subprocess
10  import concurrent.futures
11  import webbrowser
12  from datetime import datetime
```

รูปที่ 1 Library ที่ใช้งาน

2.2 Config

2.2.1 TELNET_HOST_LIST รายการ IP ของ Jump Host ที่จะใช้เข้า Telnet เพื่อเชื่อมต่อไปยังอุปกรณ์ปลายทางอีกที

2.2.2 TELNET_USER และ TELNET_PASS Username และ Password ที่ใช้ login เข้า Telnet
ใช้ในการ tn.write() เพื่อผ่านหน้าจอ username/password

2.2.3 SSH_IP_LIST = [] เป็นลิสต์ว่างไว้ก่อน รอให้ผู้ใช้โหลดไฟล์ IP เป้าหมายจาก .txt ผ่าน GUI แต่ละ IP คืออุปกรณ์ที่เราจะ SSH เข้าไป (ผ่าน jump host)

2.2.4 SSH_USER และ SSH_PASS ชื่อผู้ใช้และรหัสผ่านสำหรับ SSH ไปยังอุปกรณ์ปลายทาง หลังจาก Telnet เข้า jump host แล้ว จะใช้คำสั่ง ssh -l USER IP และใส่รหัสผ่านนี้

2.2.5 TFTP_SERVER = "10.223.255.255" กำหนด IP ของ TFTP Server ที่ใช้สำหรับ backup และ restore config ผู้ใช้สามารถ เปลี่ยนค่าได้ใน GUI ถ้า TFTP อยู่คนละเครื่องหรือ subnet

```
# --- CONFIG ---
TELNET_HOST_LIST = """
172.28.130.46
172.28.119.94
172.30.37.102
172.28.108.62
""".strip().splitlines()
TELNET_USER = "csocgov"
TELNET_PASS = "csocgov.nt"

SSH_IP_LIST = []

SSH_USER = "csocgov"
SSH_PASS = "csocgov.nt"
TFTP_SERVER = "10.223.255.255"  # default, user can overwrite in GUI
```

รูปที่ 2 ส่วนการ config

2.3 GUI set up และ UI ELEMENTS

2.3.1 GUI Setup สร้างหน้าต่างหลักของโปรแกรม (root window) ด้วย tkinter,

ตั้งชื่อหน้าต่าง (แสดงบน Title bar),กำหนดขนาดเริ่มต้นของหน้าต่างกว้าง 1100px สูง 900px

2.3.2 UI ELEMENTS สร้างกรอบสำหรับวางปุ่ม control หลัก เช่น Start, Export, Load IP ใช้ .pack(pady=10) เพื่อเว้นระยะห่างแนวตั้ง 10 px

2.3.3 ฟังก์ชัน open_output_folder() ฟังก์ชันนี้จะเปิด โฟลเดอร์ที่เก็บไฟล์สรุป .csv โดยใช้ default file explorer (เช่น File Explorer บน Windows)

2.3.4 ฟังก์ชัน load_ip_list() เปิด dialog ให้ผู้ใช้เลือกไฟล์ .txt ที่มีรายการ IP ของอุปกรณ์ที่ต้องการสำรอง

```python
# --- GUI Setup ---
root = tk.Tk()
root.title("TFTP Auto Backup Dashboard")
root.geometry("1100x900")

# --- UI ELEMENTS ---
btn_frame = tk.Frame(root)
btn_frame.pack(pady=10)

def open_output_folder():
    webbrowser.open(os.path.abspath(os.path.dirname(SUMMARY_FILE)))

def load_ip_list():
    global SSH_IP_LIST
    file_path = filedialog.askopenfilename(title="Select IP List File", filety
    if file_path:
        with open(file_path, 'r') as f:
            SSH_IP_LIST = [line.strip() for line in f if line.strip()]
        ip_status_label.config(text=f"✅ Loaded {len(SSH_IP_LIST)} IPs from fi
    else:
        ip_status_label.config(text="⚠ No file selected")

def check_tftp_server():
    ip = tftp_entry.get().strip()
    reachable = is_pingable(ip)
    if reachable:
        tftp_status_label.config(text=f"🟢 TFTP {ip} is reachable", fg="green"
    else:
        tftp_status_label.config(text=f"🔴 TFTP {ip} unreachable", fg="red")
```

รูปที่ 3 GUI set up และ UI ELEMENTS

2.4 ปุ่มควบคุม GUI

2.4.1 สร้างปุ่ม “Start Backup” ใช้ threading.Thread(...) เพื่อเรียก run_backup() แบบ background (GUI จะไม่ค้าง)

2.4.2 ปุ่มสำหรับเปิดโฟลเดอร์เก็บผลลัพธ์ .csv

2.4.3 ปุ่มสำหรับโหลดรายการ IP จากไฟล์ .txt

2.4.4 วางปุ่มทั้งหมดเรียงจากซ้ายไปขวา และเว้นระยะห่างระหว่างปุ่ม

```python
btn_start = tk.Button(btn_frame, text="▶ Start Backup", font=("Segoe UI", 11), command=lambda: threading.Thread(target=run_backup).start())
btn_start.pack(side=tk.LEFT, padx=5)

btn_export = tk.Button(btn_frame, text="📄 Export Result", font=("Segoe UI", 11), command=open_output_folder)
btn_export.pack(side=tk.LEFT, padx=5)

btn_load_ip = tk.Button(btn_frame, text="📂 Load IP List (txt)", font=("Segoe UI", 11), command=load_ip_list)
btn_load_ip.pack(side=tk.LEFT, padx=5)

ip_status_label = tk.Label(root, text="", font=("Segoe UI", 10), fg="blue")
ip_status_label.pack(pady=2)
```

รูปที่ 4 ปุ่มควบคุม GUI

2.5 TFTP Server Input , Summary Section , Device Backup Results Table , Raw Shell Output Log

2.5.1 TFTP Server Input สร้างเฟรมย่อยเพื่อจัดวางช่องกรอก TFTP Server และปุ่ม "Check" ช่องกรอก IP Address ของ TFTP Server

2.5.2 Summary Section สร้างกล่องสรุปผลการ backup เช่นจำนวน success, failed, skipped ช่องข้อความที่แสดงผลลัพธ์ summary แบบ real-time (อัปเดตตลอดเวลาขณะ backup) แสดงเวลาที่ผ่านไปตั้งแต่เริ่มกด "Start Backup"อัปเดตแบบนาที:วินาที ผ่านฟังก์ชัน update_time_monitor()

2.5.3 Device Backup Results Table กล่องที่ใส่ตารางสรุปผลของแต่ละ IP แยกรายตัว กำหนดหัวตาราง 5 คอลัมน์:**IP**: ที่อยู่ของอุปกรณ์ , **Ping**: Online / Offline ,**Status**: SUCCESS / FAILED / SKIPPED , **Error**: รายละเอียดข้อผิดพลาด (ถ้ามี) ,**Hostname**: ชื่อของอุปกรณ์ที่ SSH เข้าสำเร็จ

2.5.4 Raw Shell Output Log กล่องข้อความขนาดใหญ่พร้อม scrollbar สำหรับแสดง raw log จาก Telnet/SSH เช่น ข้อความการ login, การสั่ง copy config, ข้อผิดพลาด

```
# --- TFTP Server Input ---
tftp_frame = tk.Frame(root)
tftp_frame.pack(pady=(0, 5))

tk.Label(tftp_frame, text="TFTP Server:", font=("Segoe UI", 10)).pack(side=tk.LEFT)
tftp_entry = tk.Entry(tftp_frame, font=("Segoe UI", 10), width=30)
tftp_entry.insert(0, TFTP_SERVER)
tftp_entry.pack(side=tk.LEFT, padx=5)

btn_check_tftp = tk.Button(tftp_frame, text="🔍 Check", font=("Segoe UI", 9), command=check_tftp_server)
btn_check_tftp.pack(side=tk.LEFT, padx=5)

tftp_status_label = tk.Label(root, text="", font=("Segoe UI", 10))
tftp_status_label.pack()

summary_box = tk.LabelFrame(root, text="📊 Summary", padx=10, pady=5)
summary_box.pack(fill="x", padx=10)


summary_text = tk.Label(summary_box, justify=tk.LEFT, anchor="w", font=("Segoe UI", 10))
summary_text.pack(fill="x")
time_label = tk.Label(summary_box, text="⏱ Elapsed Time: 00:00", font=("Segoe UI", 10))
time_label.pack(side=tk.RIGHT, padx=10)

result_frame = tk.LabelFrame(root, text="📋 Device Backup Results")
result_frame.pack(fill="both", expand=True, padx=10, pady=10)

columns = ("IP", "Ping", "Status", "Error", "Hostname")
tree = ttk.Treeview(result_frame, columns=columns, show="headings")
for col in columns:
    tree.heading(col, text=col, anchor="center")
    tree.column(col, width=200 if col != "Error" else 300, anchor="center")
tree.pack(fill="both", expand=True, anchor="center")

shell_box = scrolledtext.ScrolledText(root, wrap=tk.WORD, height=10, font=("Consolas", 9), bg="#111", fg="#0f0")
shell_box.pack(fill="x", padx=10, pady=(0,10))
shell_box.insert(tk.END, "===== Telnet/SSH Raw Shell Output =====\n")
shell_box.config(state=tk.DISABLED)
```

รูปที่ 5 GUI สำหรับ TFTP Server Input , Summary Section , Device Backup Results Table , Raw Shell Output Log

2.6 Function หลักในการใช้งาน

2.6.1 is_pingable(ip) ใช้คำสั่ง ping เพื่อตรวจสอบว่า IP ปลายทาง ออนไลน์ หรือไม่mคืนค่า True ถ้า ping ได้mคืนค่า False ถ้า ping ไม่ได้
(รองรับทั้ง Windows และ Linux/Mac)

2.6.2 log_output(text) แสดงข้อความ (log) ไปที่กล่อง shell_box แบบ scroll ได้ ใช้สำหรับแสดงสถานะ Telnet/SSH, คำสั่งที่ส่ง, และผลลัพธ์จากอุปกรณ์ รองรับการอัปเดตแบบ real-time ขณะรันคำสั่ง

2.6.3 connect_and_backup_via_telnet(ip)เชื่อมต่อไปยัง jump host (Telnet) จากนั้น SSH ไปยัง IP ปลายทาง สั่ง copy running-config tftp: เพื่อสำรองค่าการตั้งค่า
คืนค่าสถานะว่า SUCCESS หรือ FAILED พร้อม error message (ถ้ามี)

2.6.4 export_results(...) บันทึกผลลัพธ์ทั้งหมดลงไฟล์ .csvรายละเอียดของแต่ละ IP: ping, status, error, hostname บันทึกสรุปท้ายไฟล์ (เช่น total, success, failed, skipped) แสดง path ของไฟล์ที่ export เสร็จใน log

2.6.5 update_time_monitor(start_time) นับเวลาที่ใช้ตั้งแต่เริ่มกด “Start Backup”

2.6.6 run_backup() ทำการสำรองค่าการตั้งค่าจากหลายอุปกรณ์พร้อมกัน อ่านรายการ IP จาก SSH_IP_LIST ใช้ thread pool รันแต่ละ IP แบบ async ตรวจสอบสถานะด้วย ping แสดงผลในตาราง GUI และ summary บันทึก .csv เมื่อเสร็จ

2.6.7 run_restore() ทำการกู้คืนค่าการตั้งค่าบนอุปกรณ์ IP เดียวเชื่อมต่อ Telnet ไปยังอุปกรณ์ปลายทางสั่ง copy tftp: running-config\กรอก TFTP IP + ชื่อไฟล์ที่เลือกจาก GUI แสดงผลลัพธ์ log และตรวจสอบว่า restore สำเร็จหรือไม่

2.6.8 ping_restore_device() ตรวจสอบว่าอุปกรณ์ที่จะ restore สามารถ ping ได้หรือไม่ แสดงผลลัพธ์ใน log ว่า Online หรือ Offline

```python
def is_pingable(ip):
    param = "-n" if platform.system().lower() == "windows" else "-c"
    # ✅ ปิดทั้ง stdout และ stderr ไม่ให้เปิด console หรือโผล่ข้อความ
    with open(os.devnull, 'w') as DEVNULL:
        result = subprocess.call(["ping", param, "1", ip],
                                 stdout=DEVNULL,
                                 stderr=DEVNULL,
                                 creationflags=subprocess.CREATE_NO_WINDOW if platform.system().lower() == "windows" else 0)
    return result == 0
def log_output(text):
    shell_box.config(state=tk.NORMAL)
    shell_box.insert(tk.END, text + "\n")
    shell_box.see(tk.END)
    shell_box.config(state=tk.DISABLED)
    root.update_idletasks()

def connect_and_backup_via_telnet(ip):
    MAX_SSH_RETRY = 1
    current_tftp = tftp_entry.get().strip()
    for telnet_host in TELNET_HOST_LIST:
        try:
            log_output(f"[Telnet→SSH] Trying Telnet host {telnet_host} to reach {ip}")
            tn = telnetlib.Telnet(telnet_host, timeout=5)
            tn.read_until(b"username:", timeout=5)
            tn.write(TELNET_USER.encode("ascii") + b"\n")
            tn.read_until(b"Password:", timeout=5)
            tn.write(TELNET_PASS.encode("ascii") + b"\n")
            jump_prompt = tn.read_until(b"#", timeout=10).decode("utf-8", errors="ignore")
            jump_lines = jump_prompt.strip().splitlines()
            if not jump_lines:
                tn.close()
                return "FAILED", "No prompt after Telnet login", ""
            jump_host_name = jump_lines[-1].replace("#", "").strip()
            ssh_success = False
            ssh_host_name = ""
            for attempt in range(1, MAX_SSH_RETRY + 1):
                tn.write(f"ssh -l {SSH_USER} {ip}\n".encode("ascii"))
                ssh_stage_output = tn.read_until(b":", timeout=10).decode("utf-8", errors="ignore")
                if any(bad in ssh_stage_output.lower() for bad in ["translating", "% bad", "refused", "timeout"]):
                    tn.write(b"\n")
                    continue
```

รูปที่ 6 Function สำหรับ is_pingable(ip),log_output(text),connect_and_backup_via_telnet(ip)

```python
                    continue
                tn.write(SSH_PASS.encode("ascii") + b"\n")
                ssh_prompt = tn.read_until(b"#", timeout=10).decode("utf-8", errors="ignore")
                ssh_lines = ssh_prompt.strip().splitlines()
                if not ssh_lines:
                    tn.write(b"\n")
                    continue
                ssh_host_name = ssh_lines[-1].replace("#", "").strip()
                if ssh_host_name == jump_host_name:
                    tn.write(b"exit\n")
                    continue
                ssh_success = True
                break
            if not ssh_success:
                tn.close()
                return "FAILED", "SSH failed", ""
            tn.write(b"terminal length 0\n")
            tn.read_until(b"#", timeout=5)
            tn.write(b"copy running-config tftp:\n")
            tn.read_until(b"Address or name", timeout=10)
            tn.write(current_tftp.encode("ascii") + b"\n")
            tn.read_until(b"filename", timeout=10)
            tn.write(b"\n")
            output = tn.read_until(b"#", timeout=20).decode("utf-8", errors="ignore")
            tn.close()
            if "copied" in output.lower():
                return "SUCCESS", "", ssh_host_name
            return "FAILED", "No 'copied' found", ssh_host_name
        except Exception as e:
            log_output(f"[ERROR] Telnet host {telnet_host} failed: {e}")
            continue
    return "FAILED", "All Telnet hosts failed", ""
```

รูปที่ 7 Function สำหรับ connect_and_backup_via_telnet(ip)(ต่อ)

```python
def export_results(results, success_count, skip_count, online_count):
    try:
        with open(SUMMARY_FILE, "w", newline='', encoding="utf-8-sig") as f:
            writer = csv.writer(f)
            writer.writerow(["IP Address", "Ping Status", "Backup Status", "Error Detail", "SSH Hostname"])
            for row in results:
                writer.writerow(row)
            writer.writerow([])
            writer.writerow(["📋 Summary"])
            writer.writerow(["Total Devices", len(results)])
            writer.writerow(["🟢 Online Devices", online_count])
            writer.writerow(["✅ Backup Success", success_count])
            writer.writerow(["❌ Backup Failed", len(results) - skip_count - success_count])
            writer.writerow(["⏭️ Skipped Offline", skip_count])
            log_output(f"\n📄 Exported summary to: {os.path.abspath(SUMMARY_FILE)}")
    except Exception as e:
        log_output(f"[ERROR] Export failed: {e}")

def update_time_monitor(start_time):
    def update():
        if not update_time_monitor.running:
            return
        elapsed = int(time.time() - start_time)
        mins, secs = divmod(elapsed, 60)
        time_label.config(text=f"⏱ Elapsed Time: {mins:02}:{secs:02}")
        root.after(1000, update)

    update_time_monitor.running = True
    update()
```

รูปที่ 8 Function สำหรับ export_results(...),update_time_monitor(start_time)

```python
def run_backup():
    if not SSH_IP_LIST:
        ip_status_label.config(text="❌ No IPs loaded. Please load a list first.", fg="red")
        return
    start_time = time.time()
    update_time_monitor(start_time)
    btn_start.config(state=tk.DISABLED)
    results = []
    for row in tree.get_children():
        tree.delete(row)

    # ✅ เพิ่มตรงนี้
    success_count = 0
    skip_count = 0
    online_count = 0

    def task(ip):
        nonlocal success_count, skip_count, online_count  # ✅ เพื่อให้ใช้ตัวแปรด้านนอกได้
        if is_pingable(ip):
            online_count += 1
            status, error, hostname = connect_and_backup_via_telnet(ip)
            if status == "SUCCESS":
                success_count += 1
            result = (ip, "Online", status, error, hostname)
        else:
            skip_count += 1
            result = (ip, "Offline", "SKIPPED", "Host unreachable", "")
        results.append(result)
        tree.insert("", tk.END, values=result)

        # 🔁 อัปเดต summary แบบ real-time
        summary_text.config(text=(
            f"💻 Total Devices: {len(SSH_IP_LIST)}    "
            f"🟢 Online: {online_count}    "
            f"⏭️ Offline / Skip: {skip_count}    "
            f"✅ Success: {success_count}    "
            f"❌ Failed: {online_count - success_count}    "
        ))
```

รูปที่ 9 Function สำหรับ run_backup()

```
    # รันพร้อมกันหลาย task
    with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
        futures = [executor.submit(task, ip) for ip in SSH_IP_LIST]
        concurrent.futures.wait(futures)

    # export สรุปเมื่อเสร็จทุก IP
    export_results(results, success_count, skip_count, online_count)
    btn_start.config(state=tk.NORMAL)
    update_time_monitor.running = False
```

รูปที่ 10 Function สำหรับ run_backup()(ต่อ)

```
272 def run_restore():
273     ip = restore_ip_entry.get().strip()
274     config_file = restore_file_path.get().strip()
275     tftp_ip = tftp_entry.get().strip()
276 
277     if not ip or not config_file:
278         log_output("❌ Please fill in both IP and config file.")
279         return
280 
281     if not os.path.exists(config_file):
282         log_output("❌ Config file not found.")
283         return
284 
285     config_filename = os.path.basename(config_file)
286 
287     log_output("=== 🚀 Starting Restore Process ===")
288     log_output(f"📱 Target Device IP: {ip}")
289     log_output(f"📁 Selected Config File: {config_filename}")
290     log_output(f"🌐 TFTP Server: {tftp_ip}")
291 
292     try:
293         log_output(f"🔌 Connecting via Telnet to {ip}...")
294         tn = telnetlib.Telnet(ip, timeout=5)
295 
296         log_output("🔐 Logging in...")
297         tn.read_until(b"username:", timeout=5)
298         tn.write(TELNET_USER.encode("ascii") + b"\n")
299         tn.read_until(b"Password:", timeout=5)
300         tn.write(TELNET_PASS.encode("ascii") + b"\n")
301         tn.read_until(b"#", timeout=10)
302         log_output("✅ Telnet Login Success")
303 
304         log_output("⚙ Setting terminal length...")
305         tn.write(b"terminal length 0\n")
306         tn.read_until(b"#", timeout=3)
307 
308         log_output("📤 Sending restore command: copy tftp: running-config")
309         tn.write(b"copy tftp: running-config\n")
310 
```

รูปที่ 11 Function สำหรับ run_restore()

```
        tn.read_until(b"Address or name", timeout=5)
        log_output(f"📤 Sending TFTP IP: {tftp_ip}")
        tn.write(tftp_ip.encode("ascii") + b"\n")

        tn.read_until(b"filename", timeout=5)
        log_output(f"📤 Sending config filename: {config_filename}")
        tn.write(config_filename.encode("ascii") + b"\n")

        tn.read_until(b"Destination filename", timeout=5)
        log_output("📤 Confirming destination filename (Enter)")
        tn.write(b"\n")

        log_output("⏳ Waiting for operation to finish...")
        output = tn.read_until(b"#", timeout=20).decode("utf-8", errors="ignore")
        tn.close()

        log_output("📦 Device Output:")
        log_output(output.strip())

        if "copied" in output.lower():
            log_output(f"✅ Restore COMPLETE to {ip}")
        else:
            log_output(f"❌ Restore FAILED to {ip} – No 'copied' confirmation in output")

    except Exception as e:
        log_output(f"❌ ERROR during Restore to {ip}: {e}")

    log_output("=== ✅ Restore Process Finished ===\n")
```

รูปที่ 12 Function สำหรับ run_restore()(ต่อ)

```
def ping_restore_device():
    ip = restore_ip_entry.get().strip()
    if not ip:
        log_output("⚠ Please enter an IP to ping.")
        return
    if is_pingable(ip):
        log_output(f"🟢 {ip} is reachable (Ping OK)")
    else:
        log_output(f"🔴 {ip} is unreachable (Ping FAIL)")
```

รูปที่ 13 Function สำหรับ ping_restore_device()(ต่อ)

2.7 การตั้งค่ารูปแบบสีของ GUI ให้ดูทันสมัย และด้วย Dark Theme การกำหนดค่าสีหลัก , ตั้งค่าพื้นหลังของหน้าต่างหลัก , ปรับ Label สำคัญให้มี accent color , ปรับสีของกล่อง log shell_boxปรับดีไซน์,สีของช่องกรอก

```
# --- DARK MODERN THEME CONFIG ---
modern_bg = "#1e1e2f"          # พื้นหลังหลัก
modern_fg = "#f0f0f0"          # ตัวอักษรหลัก
accent_color = "#4dd0e1"       # สีหลัก (ฟ้าเขียว)
entry_bg = "#2c2c3c"           # ช่องกรอก
tree_bg = "#2a2a3a"            # ตาราง
tree_fg = "#f0f0f0"            # ตัวอักษรตาราง
highlight_color = "#00acc1"    # สีเน้นเมื่อเลือก

# ตั้งค่าพื้นหลังของ root
root.configure(bg=modern_bg)

# ปรับพื้นหลัง + ตัวอักษรทุก widget (Frame, LabelFrame, Label)
for widget in root.winfo_children():
    if isinstance(widget, (tk.Frame, tk.LabelFrame)):
        widget.configure(bg=modern_bg)
    elif isinstance(widget, tk.Label):
        widget.configure(bg=modern_bg, fg=modern_fg)


# ปรับเฉพาะ label ที่ไม่เข้ากลุ่มข้างบน
ip_status_label.config(bg=modern_bg, fg=accent_color)
tftp_status_label.config(bg=modern_bg, fg=accent_color)
summary_text.config(bg=modern_bg, fg=modern_fg)
summary_box.config(bg=modern_bg, fg=modern_fg)
result_frame.config(bg=modern_bg, fg=modern_fg)
time_label.config(fg=accent_color, bg=modern_bg)

# ปรับ scrolledtext shell box
shell_box.config(bg="#141421", fg="#80cbc4", insertbackground="white")

# ปรับปุ่มให้ดูเรียบ modern
button_list = [btn_start, btn_export, btn_load_ip, btn_check_tftp]
for btn in button_list:
    btn.config(bg="#333344", fg=modern_fg, activebackground="#444455", activeforeground=accent_color, relief=tk.FLAT)

# Entry สี
tftp_entry.config(bg=entry_bg, fg=modern_fg, insertbackground=modern_fg)
```

รูปที่ 14 ตั้งค่ารูปแบบสีของ GUI

2.8 Treeview Modern Style และ GUI Restore Section กล่อง Restore Config,Label + ช่องกรอก Restore IP ,Label + ช่องกรอกไฟล์ Config

```
# Treeview modern style
style = ttk.Style()
style.theme_use("default")
style.configure("Treeview",
                background=tree_bg,
                foreground=tree_fg,
                fieldbackground=tree_bg,
                rowheight=24,
                bordercolor=modern_bg,
                borderwidth=0)
style.configure("Treeview.Heading",
                background="#333344",
                foreground=accent_color,
                font=("Segoe UI", 10, "bold"))
style.map("Treeview",
          background=[("selected", highlight_color)],
          foreground=[("selected", "white")])
# --- Restore Section ---
restore_frame = tk.LabelFrame(root, text="🛠 Restore Config", padx=10, pady=5)
restore_frame.pack(fill="x", padx=10, pady=(5, 10))

tk.Label(restore_frame, text="Restore IP:", font=("Segoe UI", 10), bg=modern_bg, fg=modern_fg).pack(side=tk.LEFT)
restore_ip_entry = tk.Entry(restore_frame, font=("Segoe UI", 10), width=20, bg=entry_bg, fg=modern_fg, insertbackground=modern_fg)
restore_ip_entry.pack(side=tk.LEFT, padx=5)

tk.Label(restore_frame, text="Config File:", font=("Segoe UI", 10), bg=modern_bg, fg=modern_fg).pack(side=tk.LEFT)
restore_file_path = tk.StringVar()
restore_file_entry = tk.Entry(restore_frame, textvariable=restore_file_path, font=("Segoe UI", 10), width=40, bg=entry_bg, fg=modern_fg, insertbackground=modern_fg)
restore_file_entry.pack(side=tk.LEFT, padx=5)
```

รูปที่ 15 ตั้งค่า Treeview Modern Style และ GUI Restore Section

2.9 browse_config_file() ฟังก์ชันเลือกไฟล์ Config ,ปุ่มควบคุม Restore GUI Browse ปุ่มเลือกไฟล์ Start Restore ปุ่มเริ่ม Restore Ping Device ปุ่มเช็คสถานะ IP , ปิดโปรแกรมอย่างปลอดภัย + Log ข้อผิดพลาด

```
422
423 def browse_config_file():
424     file_path = filedialog.askopenfilename(title="Select Config File", filetypes=[("All Files", "*.*")])
425     if file_path:
426         restore_file_path.set(file_path)
427
428 btn_browse_restore = tk.Button(restore_frame, text="📂 Browse", font=("Segoe UI", 9), command=browse_config_file)
429 btn_browse_restore.pack(side=tk.LEFT, padx=5)
430
431 btn_restore = tk.Button(restore_frame, text="📤 Start Restore", font=("Segoe UI", 10),
432                         command=lambda: threading.Thread(target=run_restore).start())
433 btn_restore.pack(side=tk.LEFT, padx=5)
434 btn_ping_restore = tk.Button(restore_frame, text="🔍 Ping Device", font=("Segoe UI", 9),
435                              command=lambda: ping_restore_device())
436 btn_ping_restore.pack(side=tk.LEFT, padx=5)
437
438 try:
439     root.mainloop()
440 except Exception as e:
441     with open("gui_error.log", "w") as f:
442         f.write(str(e))
443
```

รูปที่ 16 ตั้งค่าbrowse_config_file() , ปุ่มควบคุม Restore GUI, Log ข้อผิดพลาด

3.คู่มือการใช้งานโปรแกรม

3.1 การ Backup อัตโนมัติ

3.1.1 กด Load IP List แล้วเลือกไฟล์ .txt ตามรูปที่ 18 ที่มีรายการ IP ของอุปกรณ์เป้าหมาย

3.1.2 ตรวจสอบ TFTP Server ว่า ping ได้ด้วยปุ่ม Check

3.1.3 คลิก Start Backup

3.1.4 รอดูผลลัพธ์เรียลไทม์ที่ตารางและ shell log

3.1.5 เมื่อเสร็จ กด Export Result เพื่อเปิดไฟล์สรุป .csv

3.2 การ Restore การตั้งค่า

3.2.1 กรอก IP ของอุปกรณ์

3.2.2 กด Browse เพื่อเลือกไฟล์ config ต้องอยู่ใน folder TFTP

3.2.3 ตรวจสอบว่า TFTP Server กรอกถูกต้อง

3.2.4 กด Start Restore

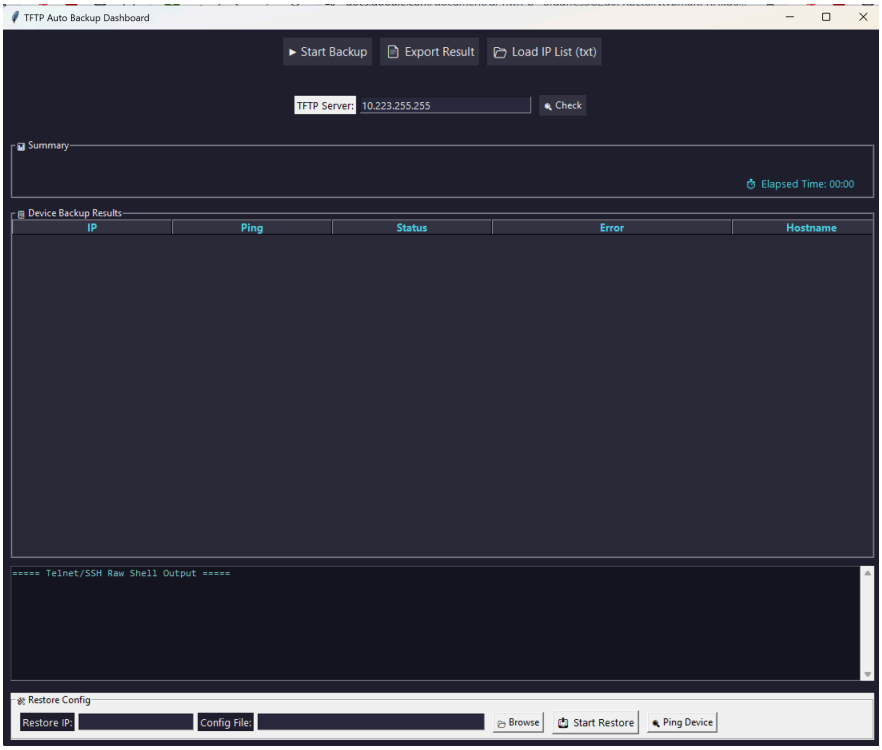
3.2.5 ดูผลการ restore จาก shell log ด้านล่าง

3.3 Output ที่ได้หลังใช้งาน

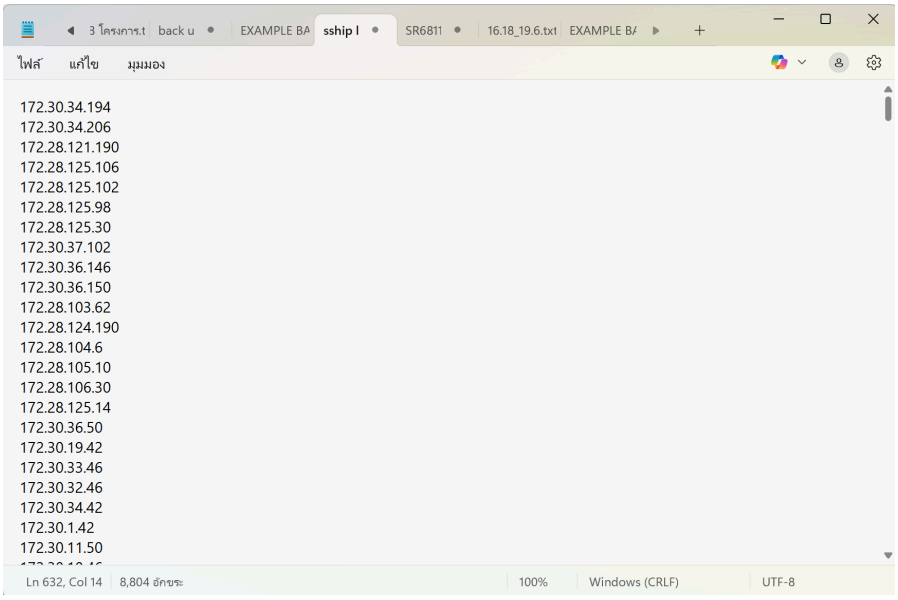
3.3.1.ไฟล์ .csv สรุปผลการ Backup

3.3.2 อยู่ในโฟลเดอร์output /backup_SW_summary_YYYYMMDD_HHMMSS.csv

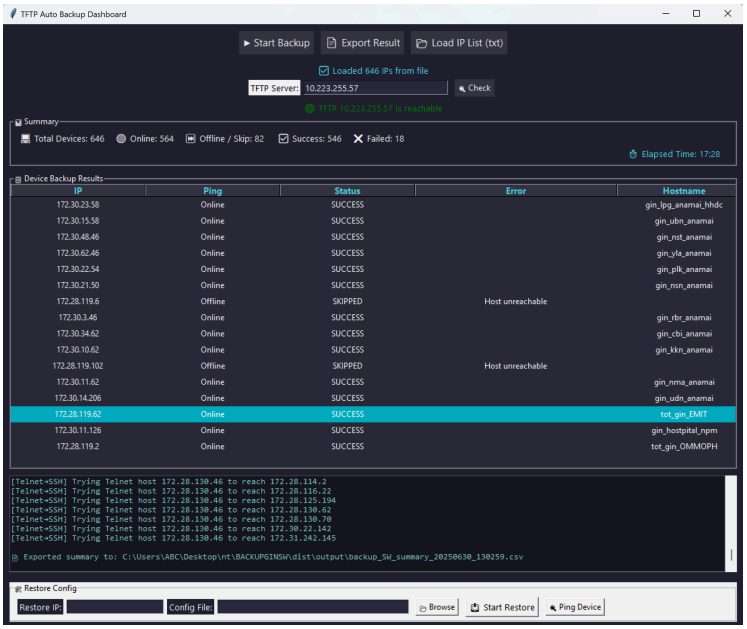
3.3.3.บรรจุ:รายการ IP ผล ping ผล Backup ข้อผิดพลาด Hostname



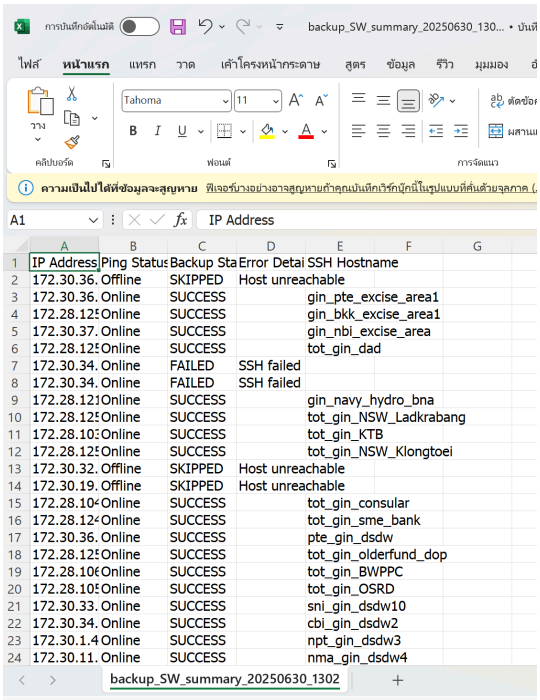
รูปที่ 17 หน้าต่างโปรแกรม Backup และ Restore



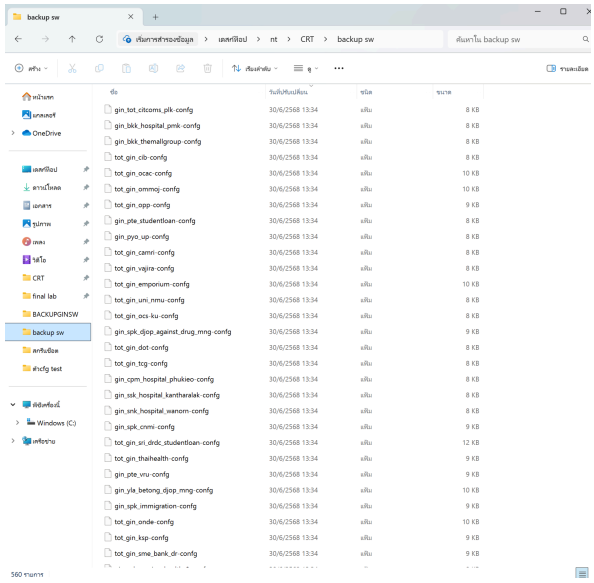
รูปที่ 18 Load IP List (.txt)



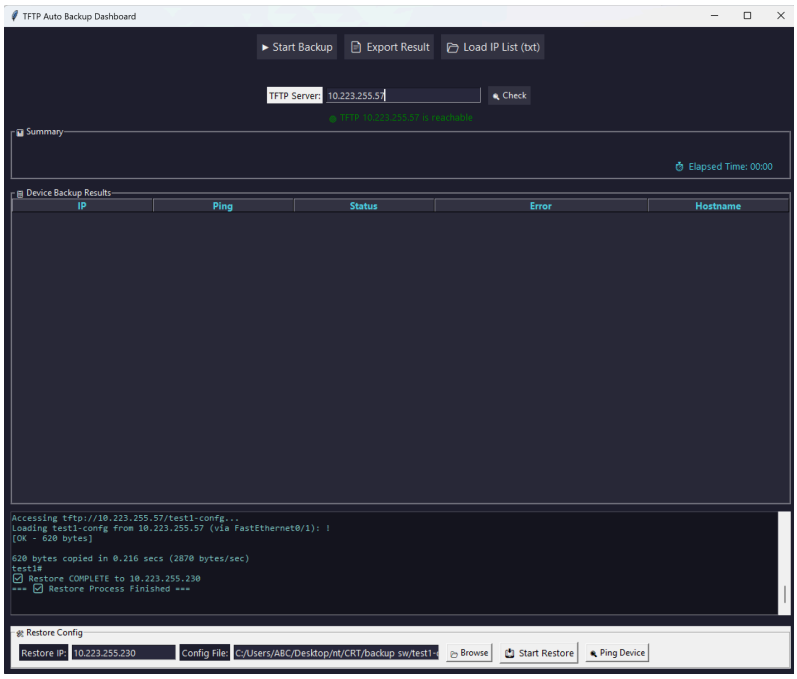
รูปที่ 19 ผลลัพธ์เมื่อ Backup ข้อมูลครบในโปรแกรม



รูปที่ 20 ผลลัพธ์เมื่อ Backup ข้อมูลครบในCSV



รูปที่ 21 ผลลัพธ์เมื่อ Backup ข้อมูลครบในTFTP Server



รูปที่ 22 ผลลัพธ์เมื่อ Restore ข้อมูลครบในโปรแกรม