

ReconAutomator GUI: A Unified Reconnaissance Tool for Cybersecurity Analysis

Your Full Name : uday marupaka

instructor : febin sir

Date : 02/05/202

Abstract

ReconAutomator GUI+ is an enhanced cybersecurity reconnaissance tool designed to streamline and centralize the process of information gathering from a target IP or domain. With added tools and modern reporting features, it supports both aggressive and stealth scanning modes, enabling cybersecurity professionals, CTF participants, and learners to conduct effective and visually accessible recon with ease.

1. Introduction

Reconnaissance is a foundational step in ethical hacking and vulnerability assessment. ReconAutomator GUI+ improves upon traditional recon practices by combining multiple tools in a GUI-based interface. This eliminates the need for switching between terminals or manually chaining tool outputs.

2. Objective

- To develop an advanced GUI tool that automates information gathering from a domain/IP.
- To support **Aggressive** and **Stealth** scanning modes.
- To display **all tool outputs** in one unified interface.
- To offer **report saving** functionality in both **TXT** and **HTML** formats.

- To support threading for a responsive user experience.

3. Tools Integrated

Tool	Purpose
Nmap	Port scanning and service detection
Gobuster	Directory and file brute-forcing
Sublist3r	Subdomain enumeration
WPScan	WordPress-specific vulnerability enumeration

4. Modes of Operation

Mode	Description
Aggressive	Uses high-speed, detailed scanning options for comprehensive enumeration.
Stealth	Uses minimal scan rates and less noise to avoid detection.

5. Procedure

Code Summary

- Written in Python 3 using `tkinter`, `subprocess`, and `threading`.
- Output can be saved in HTML (styled) or TXT format.

Setup Steps

1. Create the script

```
nano greconauto.py
```

- ##### 2. Paste the complete code
- and save using `Ctrl + O`, then exit with `Ctrl + X`.

3. Install Dependencies

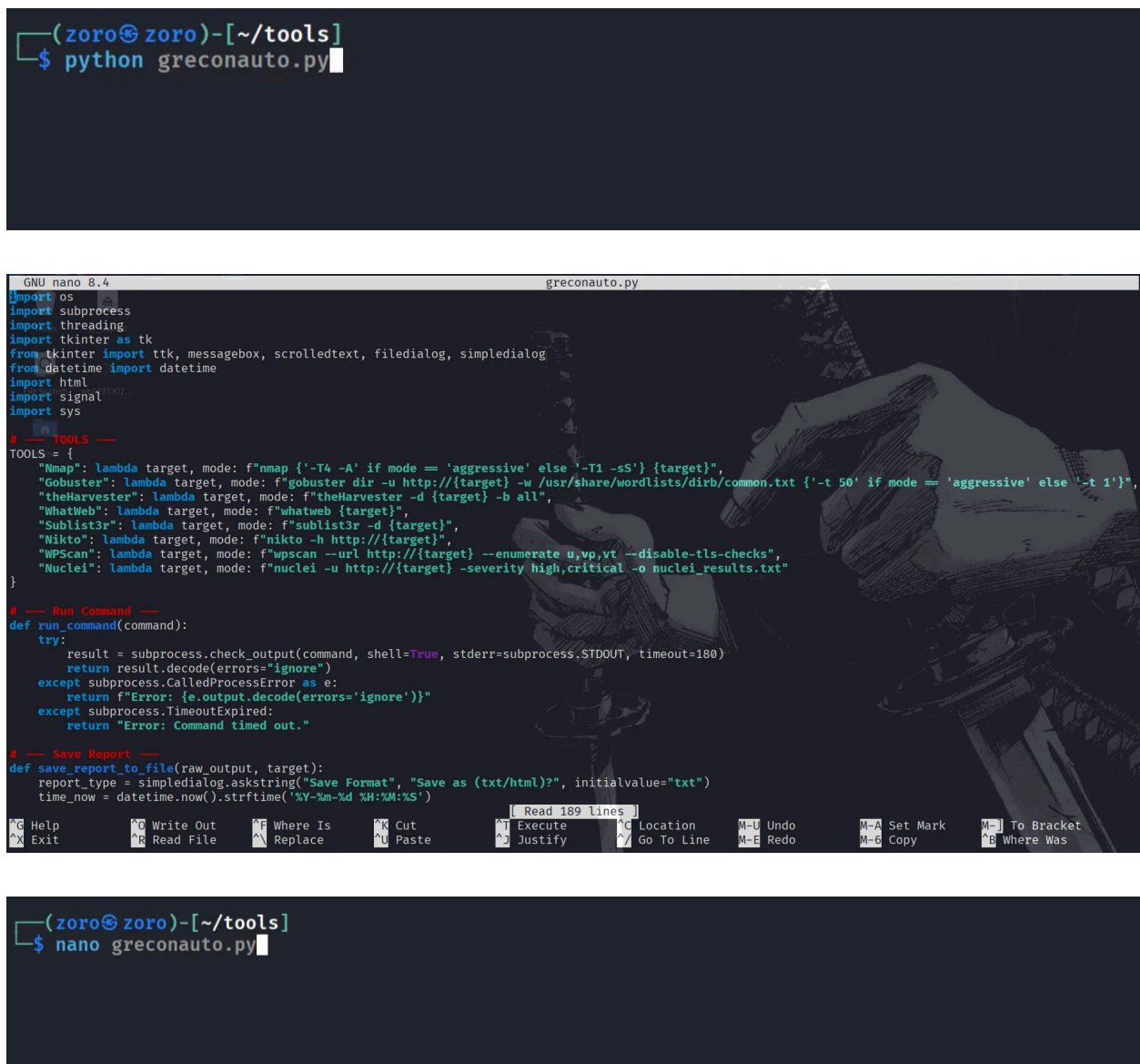
```
sudo apt update  
sudo apt install nmap gobuster whatweb sublist3r theharvester nikto wpsc
```

```
an nuclei
sudo apt install python3-tk
```

4. Run the Program

```
python3 greconauto.py
```

6.screenshots



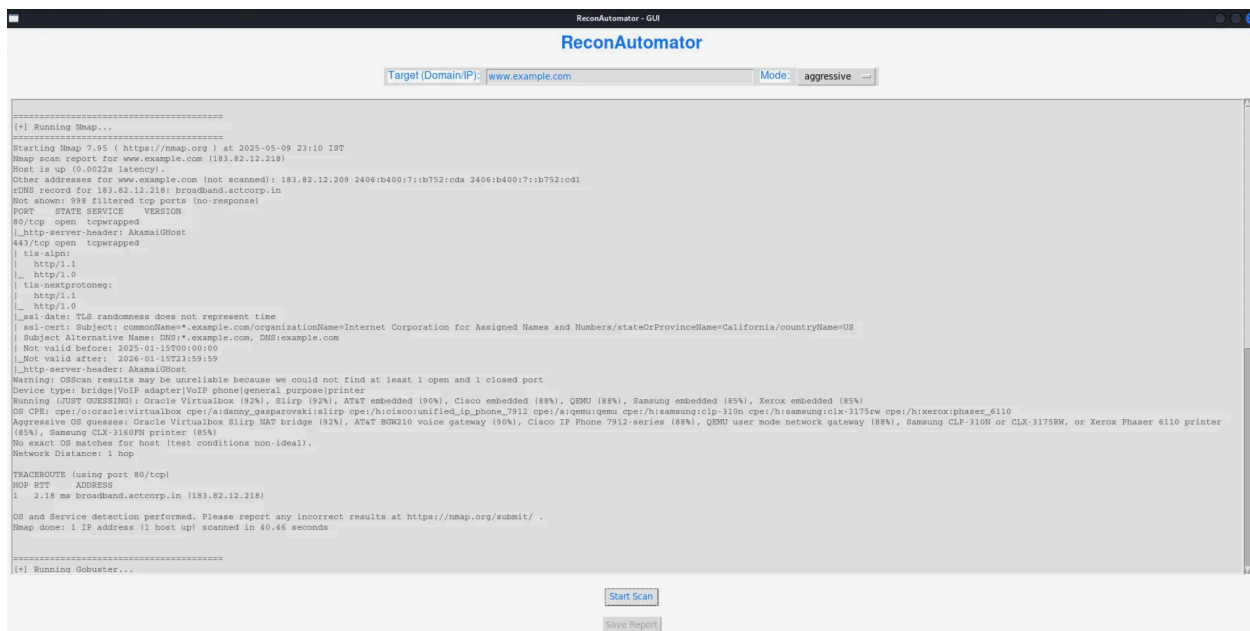
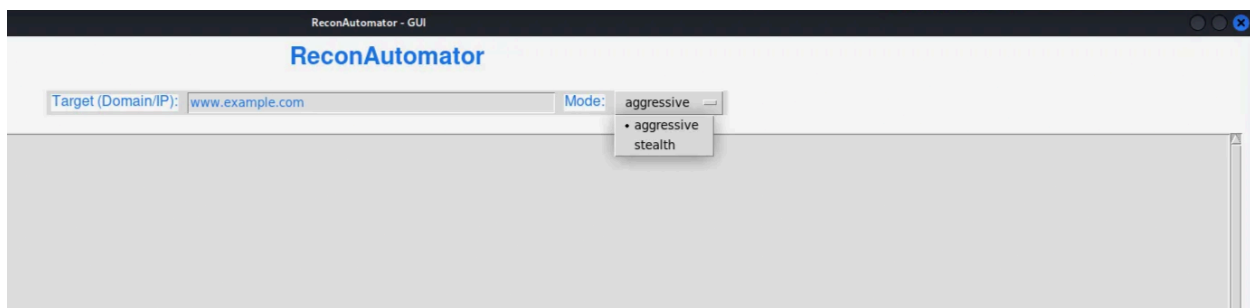
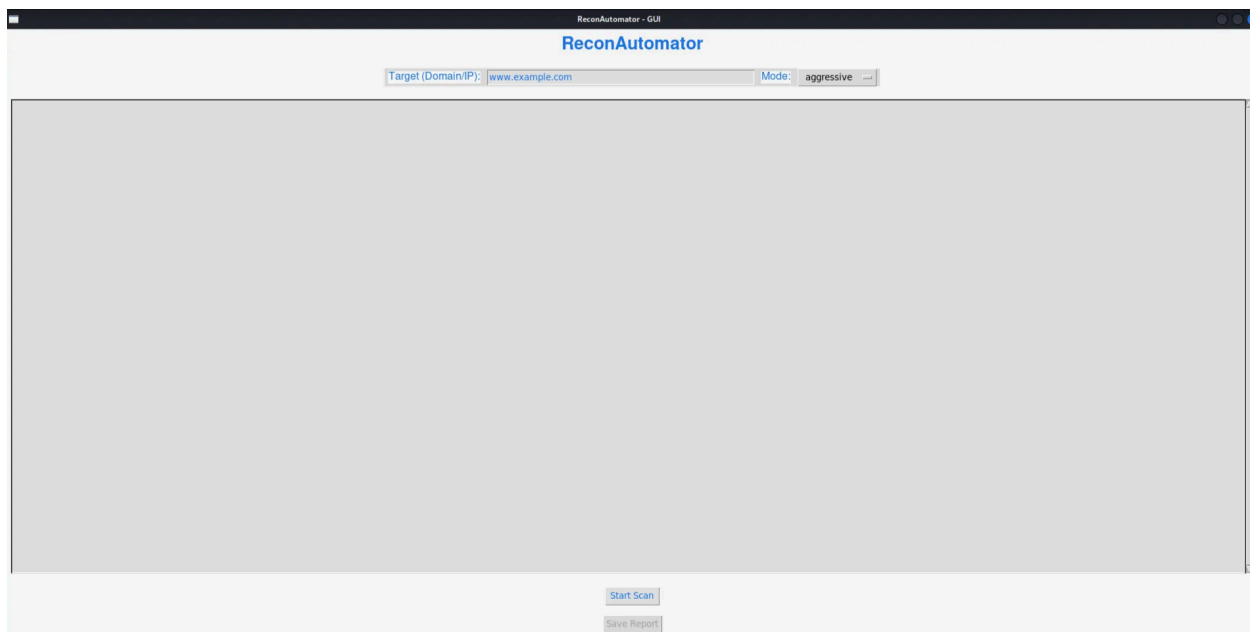
```
(zoro@zoro)-[~/tools]
$ python greconauto.py

GNU nano 8.4 greconauto.py
import os
import subprocess
import threading
import tkinter as tk
from tkinter import ttk, messagebox, scrolledtext, filedialog, simpledialog
from datetime import datetime
import html
import signal
import sys

# --- TOOLS ---
TOOLS = {
    "Nmap": lambda target, mode: f"nmap {'-T4 -A' if mode == 'aggressive' else '-T1 -sS'} {target}",
    "Gobuster": lambda target, mode: f"gobuster dir -u http://{target} -w /usr/share/wordlists/dirb/common.txt {'-t 50' if mode == 'aggressive' else '-t 1'}",
    "theHarvester": lambda target, mode: f"theHarvester -d {target} -b all",
    "WhatWeb": lambda target, mode: f"whatweb {target}",
    "Sublist3r": lambda target, mode: f"sublist3r -d {target}",
    "Nikto": lambda target, mode: f"nikto -h http://{target}",
    "WPScan": lambda target, mode: f"wpscan --url http://{target} --enumerate u,vp,vt --disable-tls-checks",
    "Nuclei": lambda target, mode: f"nuclei -u http://{target} -severity high,critical -o nuclei_results.txt"
}

# --- Run Command ---
def run_command(command):
    try:
        result = subprocess.check_output(command, shell=True, stderr=subprocess.STDOUT, timeout=180)
        return result.decode(errors="ignore")
    except subprocess.CalledProcessError as e:
        return f"Error: {e.output.decode(errors='ignore')}"
    except subprocess.TimeoutExpired:
        return "Error: Command timed out."

# --- Save Report ---
def save_report_to_file(raw_output, target):
    report_type = simpledialog.askstring("Save Format", "Save as (txt/html)?", initialvalue="txt")
    time_now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    [ Read 189 lines ]
    ^G Help      ^O Write Out  ^F Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket
    ^X Exit      ^R Read File  ^N Replace    ^U Paste      ^J Justify    ^/_ Go To Line  M-E Redo      M-G Copy      ^B Where Was
```



7. User Interface Overview

Component	Functionality
Input Field	Enter target IP or domain
Mode Dropdown	Select between Aggressive or Stealth mode
Start Scan Button	Begins the recon process
Output Display Panel	Scrollable pane showing structured results
Save Report Button	Save output in <code>.txt</code> or <code>.html</code> format

8. Technology Stack

- **Language:** Python 3
- **Libraries:** `tkinter` , `threading` , `subprocess` , `datetime` , `html`
- **Tools Integrated:** Nmap, Gobuster, WPScan

9. Conclusion

ReconAutomator GUI+ takes the tedious process of running multiple recon tools and wraps it in a user-friendly interface. Its support for dual scanning modes and flexible output format makes it a suitable choice for cybersecurity professionals and students. Future developments may include export to PDF, integration with cloud APIs, and vulnerability scoring.

10. my opinion

always automation doesn't work. we should modify as per our requirements, sometimes we have to use individually. having proper idea on when to use automation and when to not will help a lot and give proper results.