

INFORMATION-GATHERING USING GOOGLE DORKING AND NSLOOKUP COMMAND

PROJECT MEMBERS:

- KRYSTIAN CHRUPEK ([LINKEDIN](#))
- KABIR ONI ([LINKEDIN](#))
- MAWSUMI HAQUE ([LINKEDIN](#))

WARNING!

All techniques demonstrated in this project (Google dorking, nslookup and other OSINT-related tools) are used **for educational and research purposes only!**

This project does not intend to:

- gain unauthorized access,
- violate privacy,
- or misuse any gathered information.

These tools should be used **only on systems and resources you own or have explicit permission to test.**

Any other use may be illegal and is done at the **user's own responsibility.**



PROJECT OVERVIEW

- Introduction
- Requirements
- Information gathering techniques
- Script and execution + the output
- Conclusion
- References

ABOUT THE PROJECT

The goal of this project is to create a single, simple **Python** script that demonstrates and automates two fundamental information-gathering techniques (OSINT) used in reconnaissance: **Google dorking** and **DNS queries using nslookup**. The project is purely **educational** — it shows how to lawfully and responsibly collect publicly available data to analyze an attack surface and verify exposed assets.

Concretely, the script will:

- perform **Google dork** queries (examples of operators used: cache:, allintext:, allinurl:, filetype:, inurl:, site:, related:) to locate publicly accessible pages, files, and information related to the target scope;
- run DNS queries (nslookup / DNS library in Python) to fetch key DNS records such as **MX (Mail Exchange)**, **NS (Name Server)**, **A (IPv4)** and **SOA (Start of Authority)**, enabling mapping of mail infrastructure and domain name configuration.

The script's output will be presented as a clear report (screenshots/sample outputs) showing what information each technique can reveal and what security/administration conclusions can be drawn. The project emphasizes ethical usage: all demonstrations are performed only on public resources or in controlled test environments, and no unauthorized access or misuse of collected data is intended.

REQUIREMENTS

- Python (version 3.9 or newer) – to run the script
- Windows 10/11 (Script tested on Windows)
- Internet access
- Python standard library: os (used to execute system commands from the script)

INFORMATION GATHERING TECHNIQUES

- **Google dorking**

Using advanced Google search operators (e.g. site:, filetype:, inurl:, intitle:, cache:) to discover publicly available but not easily visible resources related to a target – such as exposed documents, login pages, backups, or pages indexed by mistake. This helps map what information about the target is already on the internet.

- **DNS queries with nslookup**

Querying DNS servers to retrieve key domain records (A – IPv4 address, MX – mail servers, NS – name servers, SOA – domain authority info). This allows us to understand how the domain is configured, which servers handle mail/traffic, and to identify parts of the target's infrastructure.

ICMP – PING COMMAND

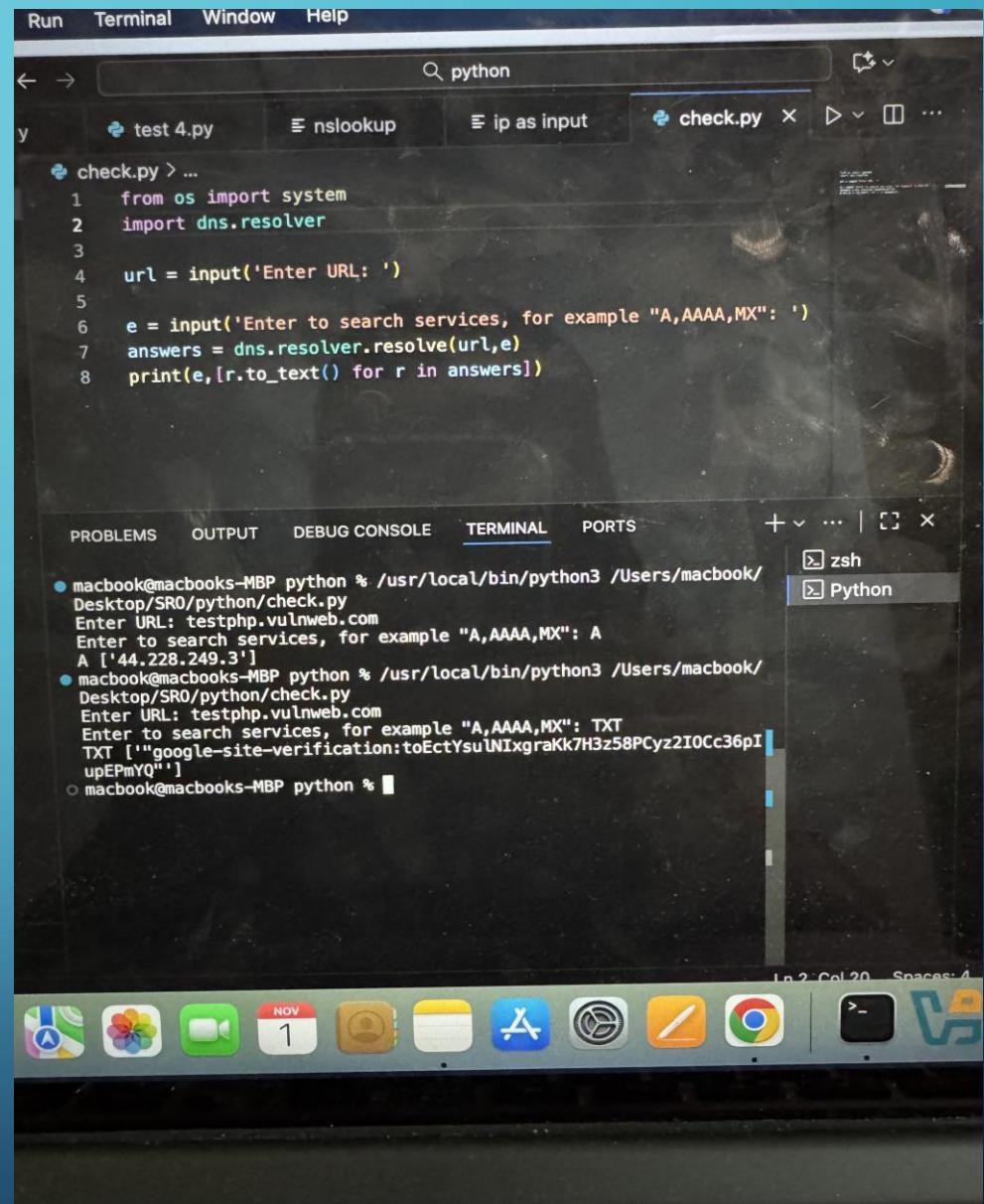
- Script execute the ICMP echo requests to the url that is taken from user, using the ping function from the system() method from os module, where the os module is a built-in Python module that provides functions for interacting with the operating system in a portable manner and system() method in Python is a powerful utility that allows you to execute shell commands directly from a Python script.

```
Go Run Terminal Window Help
... y test 4.py nslookup ip as input check.py x
check.py > ...
1   from os import system
2
3   url = input('Enter URL: ')
4
5   system("ping " + url)
6   print("Invalid Option!")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS +
macbook@macbooks-MBP python % /usr/local/bin/python3 /Users/macbook/Desktop/SR0/python/check.py
Enter URL: testphp.vulnweb.com
PING testphp.vulnweb.com (44.228.249.3): 56 data bytes
64 bytes from 44.228.249.3: icmp_seq=0 ttl=43 time=144.852 ms
64 bytes from 44.228.249.3: icmp_seq=1 ttl=43 time=132.827 ms
64 bytes from 44.228.249.3: icmp_seq=2 ttl=43 time=136.873 ms
^C
--- testphp.vulnweb.com ping statistics ---
4 packets transmitted, 3 packets received, 25.0% packet loss
round-trip min/avg/max/stddev = 132.827/138.184/144.852/4.996 ms
Invalid Option!
macbook@macbooks-MBP python % /usr/local/bin/python3 /Users/macbook/Desktop/SR0/python/check.py
Enter URL: 
```

DNS Resolver

This particular part of the script using dns.resolver library is converting human-friendly domain names into machine-readable IP addresses ,so that web browsers can connect to servers. Then **url** is taking the input and **e** is taking different DNS Records as input from user and finally store this in '**answers**' . At the end it is printing whichever services user want to check , like , A, AAAA, MX , TXT ect.

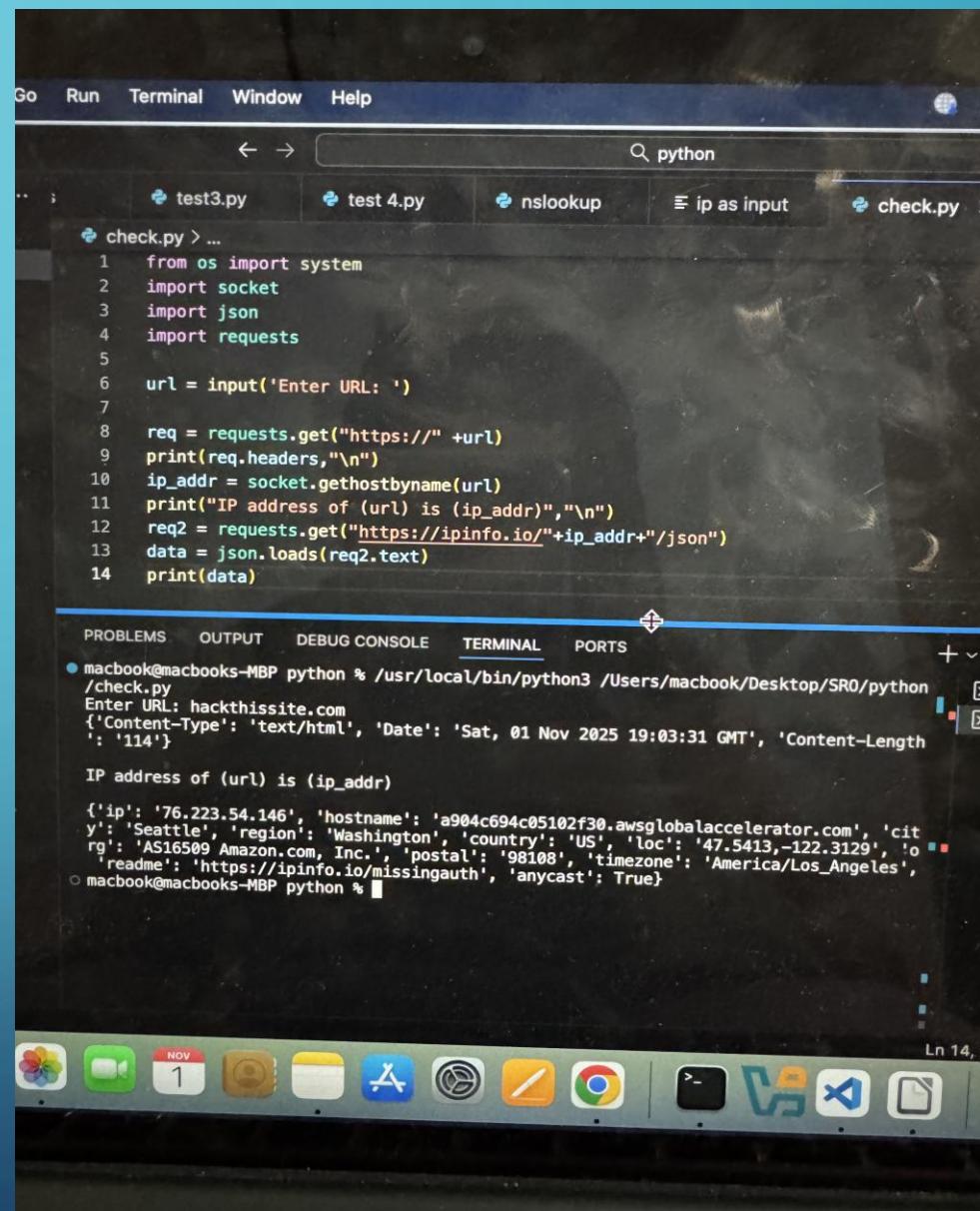


The screenshot shows a terminal window with the following details:

- Terminal Header:** Run, Terminal, Window, Help
- Search Bar:** python
- Tab Bar:** check.py (active), test 4.py, nslookup, ip as input
- Code Area:** The code for `check.py` is displayed:

```
check.py > ...
1  from os import system
2  import dns.resolver
3
4  url = input('Enter URL: ')
5
6  e = input('Enter to search services, for example "A,AAAA,MX": ')
7  answers = dns.resolver.resolve(url,e)
8  print(e,[r.to_text() for r in answers])
```
- Terminal Area:** Shows two sessions of the script being run.
 - Session 1: `macbook@macbooks-MBP python % /usr/local/bin/python3 /Users/macbook/Desktop/SR0/python/check.py`. It prompts for a URL and service types, then prints the IP address for the A record.
 - Session 2: `macbook@macbooks-MBP python % /usr/local/bin/python3 /Users/macbook/Desktop/SR0/python/check.py`. It prompts for a URL and service types, then prints the Google Site Verification TXT record.
- Bottom Bar:** Shows the Mac OS X dock with various application icons.

This particular part of the script is gathering information from inverting url in IP address by using **socket** library which connect directly with the machine. Here **requests** library used for taking the url as http:// , which is connect to the website. After converting the url to IP address by using ip_addr, req2 is using the information from ipinfo.io . Here json library used for the function Jason which is like a javascript to organise the data by calling Jason.loads.



```
check.py > ...
1  from os import system
2  import socket
3  import json
4  import requests
5
6  url = input('Enter URL: ')
7
8  req = requests.get("https://" +url)
9  print(req.headers,"\\n")
10 ip_addr = socket.gethostbyname(url)
11 print("IP address of (url) is (ip_addr)","\\n")
12 req2 = requests.get("https://ipinfo.io/" +ip_addr +"/json")
13 data = json.loads(req2.text)
14 print(data)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
macbook@macbooks-MBP python % /usr/local/bin/python3 /Users/macbook/Desktop/SR0/python
/ccheck.py
Enter URL: hackthissite.com
{'Content-Type': 'text/html', 'Date': 'Sat, 01 Nov 2025 19:03:31 GMT', 'Content-Length':
': '114'}
IP address of (url) is (ip_addr)
{'ip': '76.223.54.146', 'hostname': 'a904c694c05102f30.awsglobalaccelerator.com', 'cit
y': 'Seattle', 'region': 'Washington', 'country': 'US', 'loc': '47.5413,-122.3129', 'o
rg': 'AS16509 Amazon.com, Inc.', 'postal': '98108', 'timezone': 'America/Los_Angeles',
'readme': 'https://ipinfo.io/missingauth', 'anycast': True}
macbook@macbooks-MBP python %
```

NSLookup

Using Nslookup function from nslookup library to give the information exactly like how we can gather information from cmd/terminal command. Here, ns_query = Nslookup() initializes Nslookup, dns_query = Nslookup(dns_servers=["1.1.1.1"], verbose=False, tcp=False) this line arguments for setting custom dns servers (defaults to system DNS), verbosity (default: True) and using TCP instead of UDP (default: False). The line ips_record = dns_query.dns_lookup(domain) gets the dns information from domain, and soa_record = dns_query.soa_lookup(domain) gets the sea information from domain.

The screenshot shows a Mac OS X desktop with a terminal window open. The terminal window has tabs for 'check.py > ...', 'test3.py', 'test 4.py', 'nslookup', 'ip as input', and 'check.py'. The code in 'check.py' is:

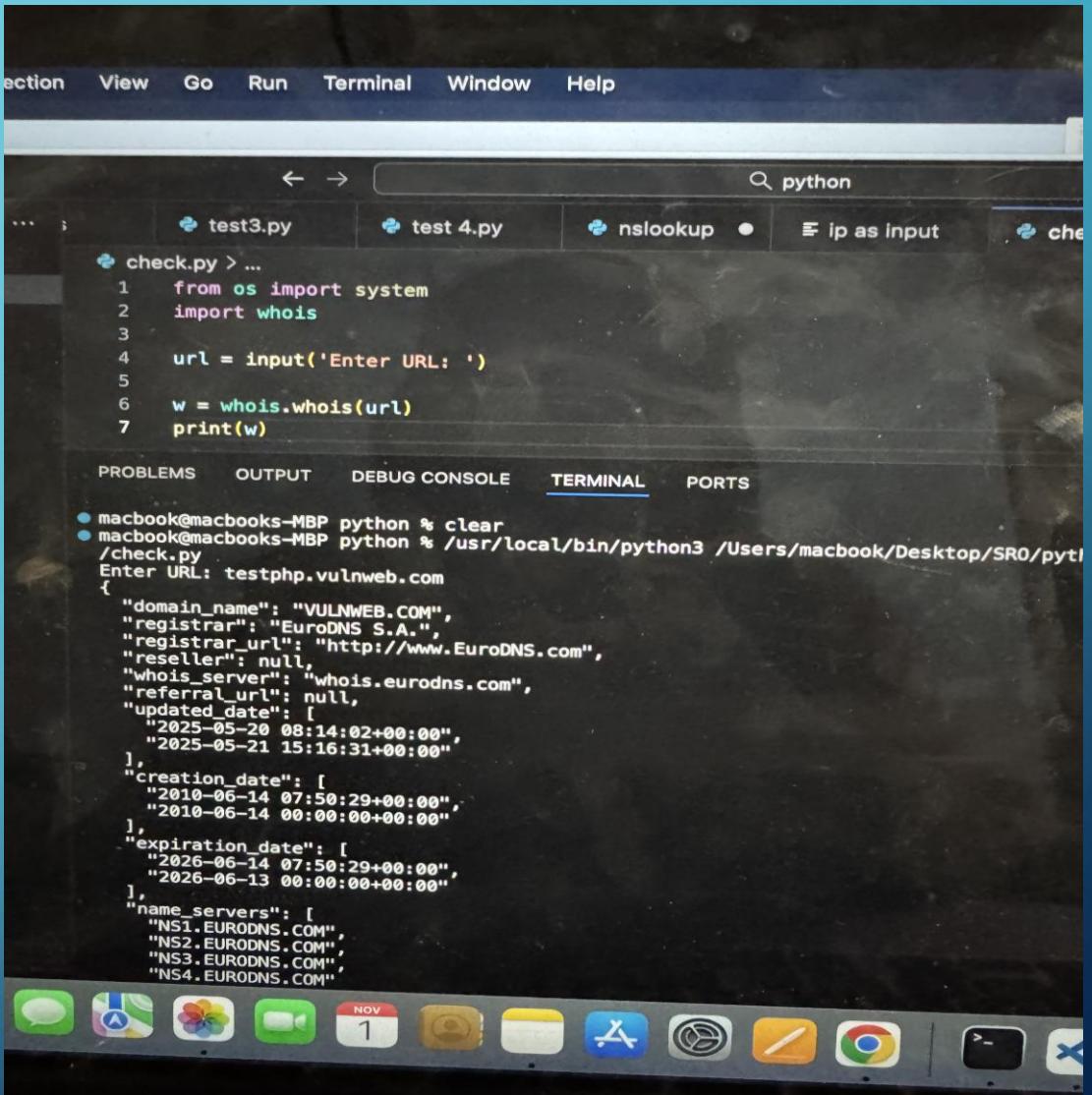
```
from os import system
from nslookup import Nslookup

url = input('Enter URL: ')
dns_query = Nslookup()
dns_query = Nslookup(dns_servers=["1.1.1.1"], verbose=False, tcp=False)
ips_record = dns_query.dns_lookup(url)
print(ips_record.response_full, ips_record.answer)
```

The terminal output shows the user entering 'testphp.vulnweb.com' and the resulting DNS response:

```
macbook@macbooks-MBP python % /usr/local/bin/python3 /Users/macbook/Desktop/SR0/python /check.py
Enter URL: testphp.vulnweb.com
['testphp.vulnweb.com. 3600 IN A 44.228.249.3'] ['44.228.249.3']
macbook@macbooks-MBP python %
```

This particular script will produce parsed WHOIS data for a given domain, Query a WHOIS server directly instead of going through an intermediate web service like many others do. There is a library called whois in python which is used to get all the information here.



The screenshot shows a Mac OS X desktop with a terminal window open in the foreground. The terminal window has tabs for "test3.py", "test 4.py", "nslookup", and "ip as input". The "check.py" file is currently being edited, showing the following Python code:

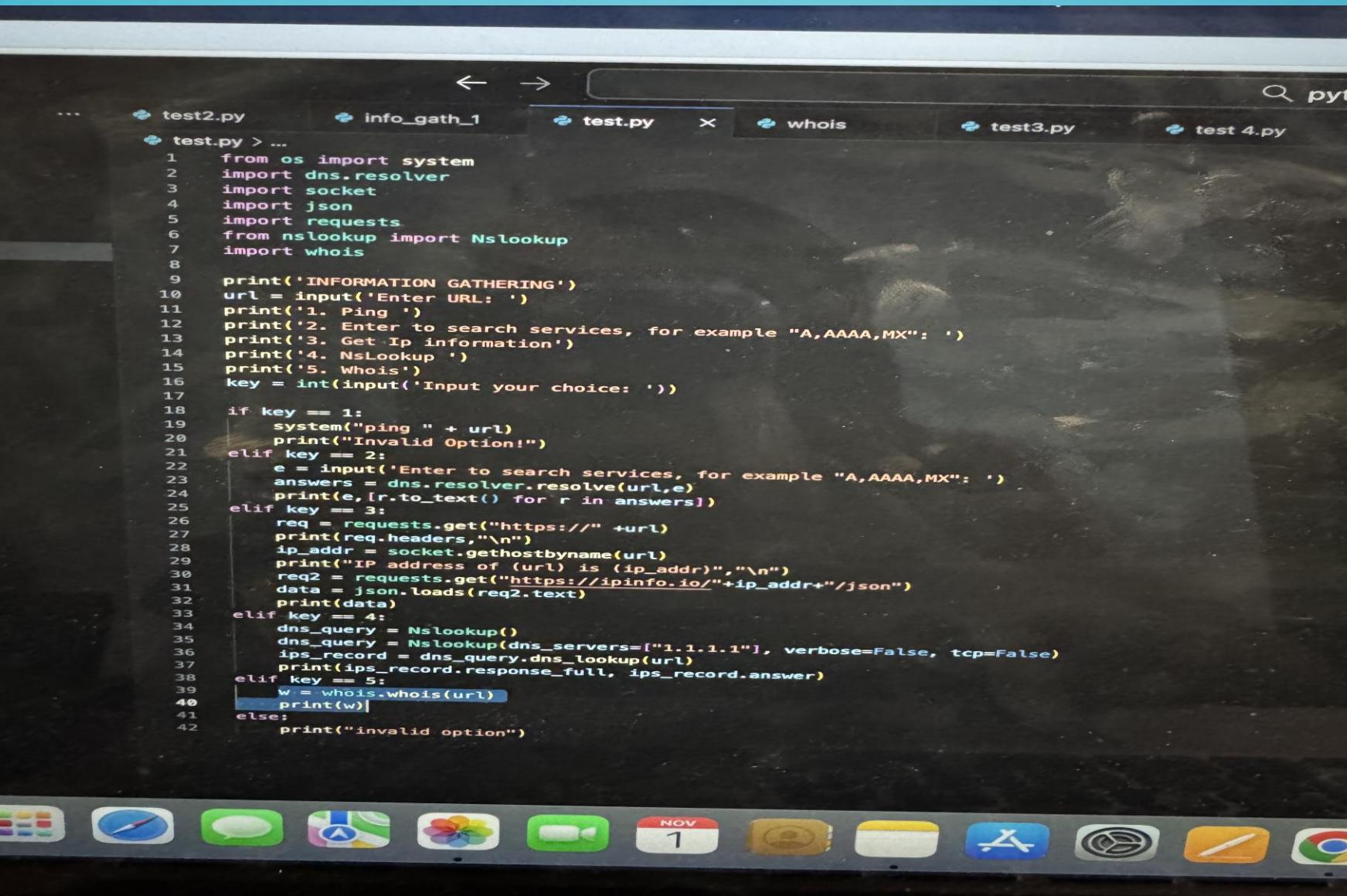
```
check.py > ...
1  from os import system
2  import whois
3
4  url = input('Enter URL: ')
5
6  w = whois.whois(url)
7  print(w)
```

Below the code editor, the terminal tab is selected, showing the command "python /check.py" and the resulting WHOIS output for the domain "testphp.vulnweb.com". The output includes details such as registration and expiration dates, name servers, and contact information.

```
macbook@macbooks-MBP:~$ clear
macbook@macbooks-MBP:~$ python /check.py
Enter URL: testphp.vulnweb.com
{
  "domain_name": "VULNWEB.COM",
  "registrar": "EuroDNS S.A.",
  "registrar_url": "http://www.EuroDNS.com",
  "reseller": null,
  "whois_server": "whois.eurodns.com",
  "referral_url": null,
  "updated_date": [
    "2025-05-20 08:14:02+00:00",
    "2025-05-21 15:16:31+00:00"
  ],
  "creation_date": [
    "2010-06-14 07:50:29+00:00",
    "2010-06-14 00:00:00+00:00"
  ],
  "expiration_date": [
    "2026-06-14 07:50:29+00:00",
    "2026-06-13 00:00:00+00:00"
  ],
  "name_servers": [
    "NS1.EURODNS.COM",
    "NS2.EURODNS.COM",
    "NS3.EURODNS.COM",
    "NS4.EURODNS.COM"
  ]}
```

The Mac OS X dock at the bottom of the screen is visible, showing icons for various applications like Mail, Finder, and Safari.

FINAL SCRITP WITH ALL OF THE COMPONENTS:



The image shows a screenshot of a Mac OS X desktop. In the foreground, a terminal window is open, displaying a Python script named 'test.py'. The script is designed for information gathering and includes imports for os, dns.resolver, socket, json, requests, nslookup, and whois. It prints a menu of five options (Ping, Services search, Get IP info, NsLookup, Whois) and takes user input to execute the selected command. The terminal window has tabs for 'test2.py', 'info_gath_1', 'test.py' (which is the active tab), 'whois', 'test3.py', and 'test 4.py'. The background of the desktop shows a blurred image of a person's face. At the bottom of the screen, the Dock contains icons for various Mac applications like Finder, Safari, Mail, and iMovie.

```
test2.py    info_gath_1    test.py    whois    test3.py    test 4.py
test.py > ...
1   from os import system
2   import dns.resolver
3   import socket
4   import json
5   import requests
6   from nslookup import Nslookup
7   import whois
8
9   print('INFORMATION GATHERING')
10  url = input('Enter URL: ')
11  print('1. Ping ')
12  print('2. Enter to search services, for example "A,AAAA,MX": ')
13  print('3. Get Ip information')
14  print('4. NsLookup ')
15  print('5. Whois')
16  key = int(input('Input your choice: '))
17
18  if key == 1:
19      system("ping " + url)
20      print("Invalid Option!")
21  elif key == 2:
22      e = input('Enter to search services, for example "A,AAAA,MX": ')
23      answers = dns.resolver.resolve(url,e)
24      print(e,[r.to_text() for r in answers])
25  elif key == 3:
26      req = requests.get("https://" +url)
27      print(req.headers,"\n")
28      ip_addr = socket.gethostname(url)
29      print("IP address of ("+url+") is ("+ip_addr+)", "\n")
30      req2 = requests.get("https://ipinfo.io/" +ip_addr+"/json")
31      data = json.loads(req2.text)
32      print(data)
33  elif key == 4:
34      dns_query = Nslookup()
35      dns_query = Nslookup(dns_servers=["1.1.1.1"], verbose=False, tcp=False)
36      ips_record = dns_query.dns_lookup(url)
37      print(ips_record.response_full, ips_record.answer)
38  elif key == 5:
39      w = whois.whois(url)
40      print(w)
41  else:
42      print("invalid option")
```

CONCLUSION

- The project demonstrated that combining two simple techniques (Google dorking and DNS/nslookup) can reveal a surprising amount of publicly available information about a domain.
- Using advanced Google operators makes it easier to reach resources that are indexed but not meant to be easily found (login pages, backups, documents, admin panels).
- DNS record analysis (A, MX, NS, SOA) allows for an initial mapping of the target's infrastructure and related services.
- Python can be used as an automation layer for reconnaissance tasks – with the os module it is possible to call system networking tools directly from the script.
- Automation must take into account limitations (search engine rate limiting, OS differences) and follow ethical rules – public data only, authorized targets only.

REFERENCES

- Mosh Hamedani, Python Full Course for Beginners (YouTube, “Programming with Mosh”)
- Python 3 Standard Library documentation (modules: os, subprocess)
- Google Search Operators – official help / support pages
- Microsoft Docs – nslookup command
- AI-assisted explanations (ChatGPT – to clarify OSINT techniques and script structure)
 - <https://www.youtube.com/watch?v=CqlW4dl8a9c>
 - <https://pypi.org/project/nslookup/>
 - <https://hackernoon.com/python-build-a-domain-lookup-tool>

