AI RED TEAMER

RESEARCHED BY:
RASHID IQBAL

"I don't want fame, I want mastery"

This is Part-1 of Documented Operations on Kali Linux

--(DOCUMENTED OPERATIONS ON KALI)--

WHAT DOES PROFESSIONALISM MEAN FOR A PENETRATION TESTER?

Professionalism for a penetration tester means doing the right thing the right way.

First, you must always follow the rules & only test systems you have permission to test. You keep everything confidential & never share what you find with anyone who shouldn't know. You stay skilled & honest about what you can & cannot do. When you report your findings, you explain them clearly so clients understand the problems and how to fix them. You show up on time, do quality work, and take responsibility for what you do. You don't make up fake problems or let anyone pressure you to lie about your results. You treat the client's systems & people with respect, stay calm when you find serious issues, and act professionally at all times.

Basically, professionalism means being trustworthy, capable, honest, and focused on helping your client in a legal and ethical way.

EVALUATE YOUR CURRENT SKILL LEVEL?

Actually, I have built a strong foundation in both IT & Cybersecurity over the past few semesters. I am very proficient in MS Office & have speed of 80 words per minute, which helps me work efficiently. I have practical experience with CCNA – Routing & Switching along with their security, I am also having proficiency of CCTV cameras along with their security perspect and I am continuously updating myself with the modern trends related to these aspects. Furthermore, I am continuously learning about network security, penetration testing, and Red Team ethical hacking.

In addition, I am exploring the Godot game engine to create small games, which I plan to integrate into cybersecurity concepts. I have structured my learning with hands-on projects, such as creating a CIA Triad-based simulated attack detection system, and I am improving my programming skills in C++ & Python.

Overall, I would describe my skill level as intermediate to advanced in IT foundation & cybersecurity basics, with a strong enthusiasm for learning & applying advanced hacking techniques and software development in practical scenarios.

MENTION ONE WEAKNESS & HOW YOU PLAN TO IMPROVE IT?

One of my most prominent weakness is that I sometimes get overly focused on technical details, which can slow me down in completing tasks quickly.

To improve this, I am actively working on better time management & prioritizing tasks, while setting small milestones to ensure steady progress without losing sight of the bigger picture.

HOW TO BUILD YOUR TESTING ENVIRONMENT (KALI LINUX, VULNERABLE LABS, BROWSERS, BURP SUITE)?

To build my testing environment I start by choosing a host machine with enough resources – at least 8 GB RAM (16+ GB preferred), a multi-core CPU and 50+ RAM GB free disk. I prefer to run virtual machines so my host stays clean: I use VirtualBox or VMware for full VMs, and sometimes WSL2 / Win-KeX for lightweight Kali access on Windows. I always download Kali Linux only from the official site & verify the ISO signature before installing to avoid tampered images.

Next I create an isolated virtual network for the lab so nothing can accidentally touch my home / office network. I use host-only or an internal NAT network in the hypervisor & enable snapshots so I can roll back any VM to a clean state after tests. I create at least two categories of VMs: Attacker (Kali) and targets (intentionally vulnerable images). Keep the lab offline or tightly NAT'd – never point vulnerable machiens to the public internet.

For target systems I use intentionally vulnerable distributions & web apps that are designed for learning: Metasploitable (v2 / v3), OWASP Juice Shop, DVWA, bWAPP, and similar vulnerable containers or VMs. I deploy these as separate VMs or Docker containers inside the isolated lab network. Use the official project repos / pages to download images & follow their install notes.

On the attacker VM (Kali) I install my toolset. For web testing I install Burp Suite (Community or Professional). I get Burp directly from PortSwigger & keep it up to date. I configure Burp to listen on a fixed proxy port (commonly 127.0.0.1:8080) and then configure my browser to use the proxy. I also import Burp's CA certificate into the browser so HTTPS traffic can be inspected without certificate errors (only inside the lab).

For browsers I create dedicated profiles for testing (separate from my normal browsing profile). I use Firefox Developer Edition or Chromium-based browsers & disable features that leak host information (extensions that sync, autofill). Each browser profile is set to use the lab proxy & has the Burp CA installed only in that profile. This keeps testing controlled & resproducible.

Operational best practices I follow:

- Always have current snapshots & backups; snapshot before any risky activity.
- Use VLANs / host-only networks so vulnerable machines can't reach the internet.
- Maintain an attack log (notes, timestamps, commands) for reproducibility & learning.
- Keep tool installs from official sources (Kali, Burp, Juice Shop repo, Metasploitable GitHub).
- Follow legal / ethical rules – only test on systems you own or have explicit permission to test.



FOLDER ORGANIZATION FOR PROJECTS (RECON, SCANS, REPORTS)?

Actually, when I manage a pentest or security project I keep a strict, predictable folder structure so work is reproducible, auditable, and easy to hand over. I create a single project root named with the client & date (for example ACME_2025-11-01) and inside it I use clear numbered folders so the order of work is obvious.

FOLDER DESCRIPTION:

FOLDER	PURPOSE	WHAT I KEEP INSIDE	NAMING / NOTES
01_Project-Info/	I keep all scope & legal artifacts	Scope.txt, Rules_of_Engagement.pdf, Contacts.csv, README.md	Project root + date; README explains structure
02_Recon/	I store raw & summarized discovery data	Subdomains.txt, whois.json, recon_notes.md, screenshots/	I separate raw outputs and a recon_notes.md summary
03_Scans/	I keep scanner outputs & parsed results	Nmap_raw.xml, nmap_parsed.csv, masscan.json, nessus_report.xml	Include scanner command + timestamp in filename
04_Exploitation/	I record PoCs & exploit artifacts	PoC_scripts/, exploit_README.md, demo_video.mp4	Each PoC has a short README & snapshot
05_Evidence/	I store immutable evidence with integrity checks	Screenshots/, captures.pcap, sha256sums.txt, logs/	Evidence is never overwritten; include hashes
06_Reports/	I draft & finalize reports & client deliverables	Exec_summary.pdf, full_report.docx, slides.pptx, changelog.md	Version files (v1, v2) and report_metadata.json
07_Templates-Tools/	I keep reusable templates, scripts, and configs	Report_template.docx, scan_commands.txt, scripts/	Use a baseline for future engagements

For file naming I use a strict, timestamped convention so everything sorts & is unambiguous.

For example:

- 2025-11-01_0930_nmap_target-10.1.1.5.xml
- 2025-11-02_1405_screenshot_login-bypass.png

I also tag sensitive files with **SENSITIVE_** and keep them encrypted at rest.

I take snapshots before risky steps, never test outside the agreed scope, keep a running log (activity_log.md) with short entries (time, action, commands, result), and produce an executive summary first so non-technical stakeholders get value immediately.

Finally, I make a final handover package that includes the final report, a sanitized evidence bundle, and a short runbook for remediation.

COMMAND-LINE DISCIPLINE AND DOCUMENTATION HABIT?

On the command line I follow strict discipline so my work is reproducible, safe, and easy to hand over. I always work in a project directory and never run risky commands from my home or random folders. I keep an activity log (activity_log.md) in the project root and log every important command I run with a short note explaining why I ran it. My log entries follow a simple, consistent format so anyone can read them later.

- [YYYY-MM-DD HH:MM] user@host: /path/to/dir - $nmap -sV target.example.com -oA scans/nmap_target – rc=0 – found: nginx 1.18 (HTTP on 80)

I enable command history timestamps in my shell (so history shows dates) and I alias a quick logger function so I never forgot to record context:

- logcmd() { echo "[$(date '+%F %T')] $PWD - $*" >> activity_log.md; eval "$@"; }

For longer sessions I record the terminal using script or asciinema so I have a full, replayable record (useful for demos and proof). I put all reusable commands and scan flags into simple scripts inside 07_Templates-Tools/ and version those scripts with Git so changes are tracked and reviewed.

My documentation habit covers three layers:

- Live log:
  Activity_log.md with timestamped command entries and short results.
- Artifacts:
  Raw output saved under 03_Scans/ and 05_Evidence/ with filenames that include timestamps and the command used.
- Narrative:
  Recon_notes.md and a short report_notes.md that explain the high-level why and next steps for non-technical readers.

I never overwrite evidence; I include the exact command used (and the tool version) when saving results; I snapshot VMs before risky action; I keep sensitive commands out of public logs (use encrypted private notes for secrets); and I use clear commit messages when I change scripts or configurations.

This discipline makes my work auditable, helps in findings quickly, and saves time during handovers or when writing reports.

KALI LINUX (LATEST VERSION)

To set up my personal lab for cybersecurity practice, I always start with a clean, organized setup & make sure everything runs in an isolated environment for safety.

First, I download the latest official Kali Linux ISO only from the offensive-security.com site and verify its SHA256 checksum to ensure the image hasn't been tampered with. Then, I install VirtualBox (or VMware Workstation) on my host system – it's safer than dual-booting because it keeps my main OS protected and lets me create isolated virtual networks.

Inside VirtualBox, I create a new virtual machine named Kali-Lab, assign at least 4-8 GB RAM, 2-4 CPU cores, and 50 GB storage. After mounting the Kali ISO, I install it normally & update it using:

```
sudo apt update && sudo apt full-upgrade -y
```

so that all packages are the latest versions.

After setup, I install key penetration testing tools if they aren't already included:

- Burp Suite (for web testing)
- Nmap and Masscan (for scanning)
- Wireshark, Hydra, Metasploit, and Gobuster
- Visual Studio Code and Python3-pip (for scripting & reports)

Then, I configure a host-only or internal NAT network in VirtualBox to ensure my lab is completely offline – meaning, any vulnerable VMs I set up (like Metasploitable2, DVWA, or OWASP Juice Shop) can't reach the internet or expose risks to my host.

I create separate folders inside my Documents directory such as:

```
~/Kali-Lab/
├── Targets/      # Vulnerable VMs
├── Reports/      # Findings & documentation
├── Scans/        # Nmap, Nikto, etc
└── Tools/        # Custom scripts & payloads
```

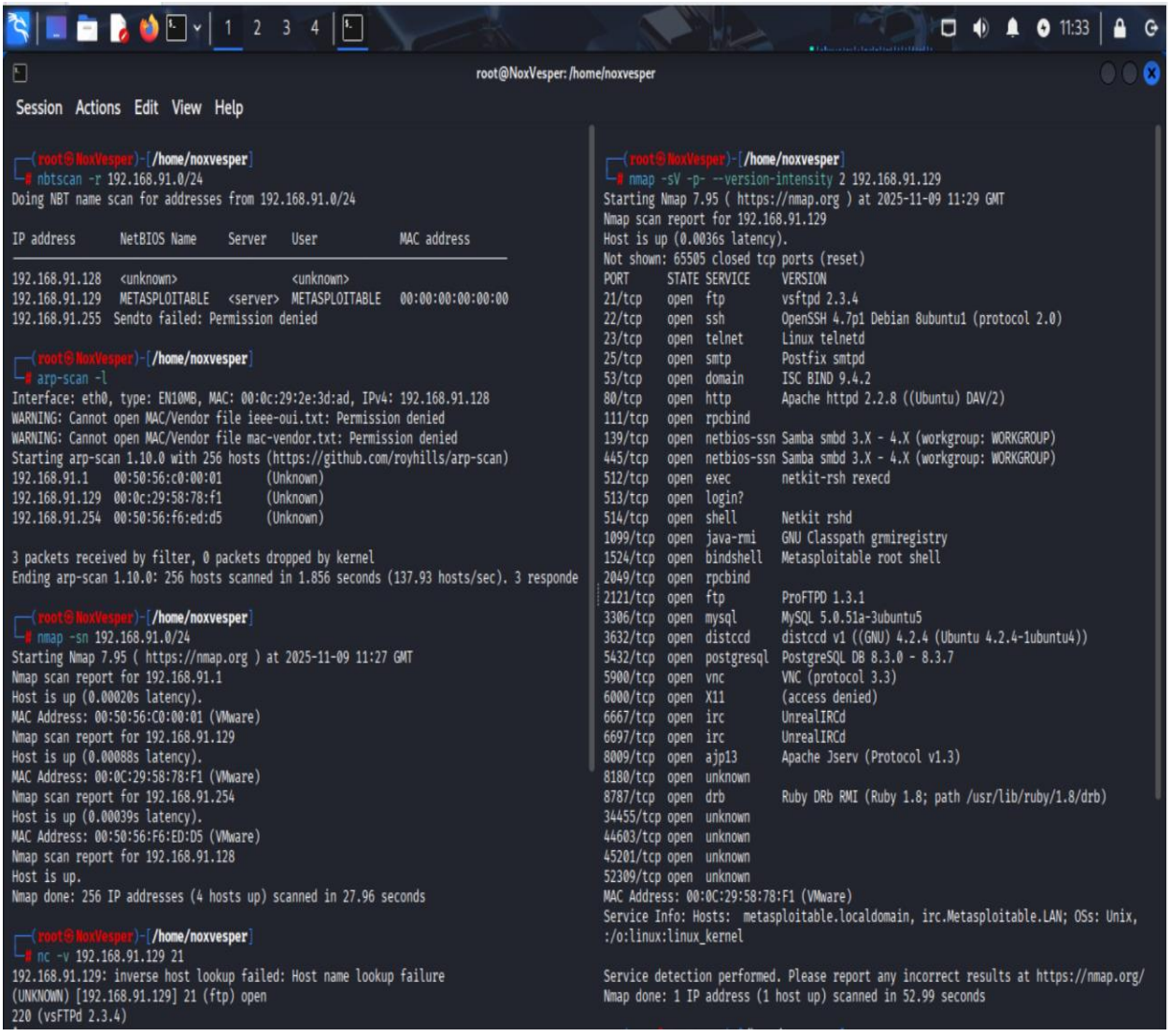
I also set snapshots in VirtualBox before testing, so I can always roll back if anything goes wrong. Finally, I secure my lab by disabling shared folders, turning off clipboard sharing, and keeping the entire environment offline during practice sessions.

This setup gives me a safe, professional-grade Kali Linux lab where I can freely practice scanning, exploitation, and reporting without affecting my main system or network.

CREATE A FOLDER STRUCTURE FOR RECON, SCANS, REPORTS FOR METASPLOITABLE2?

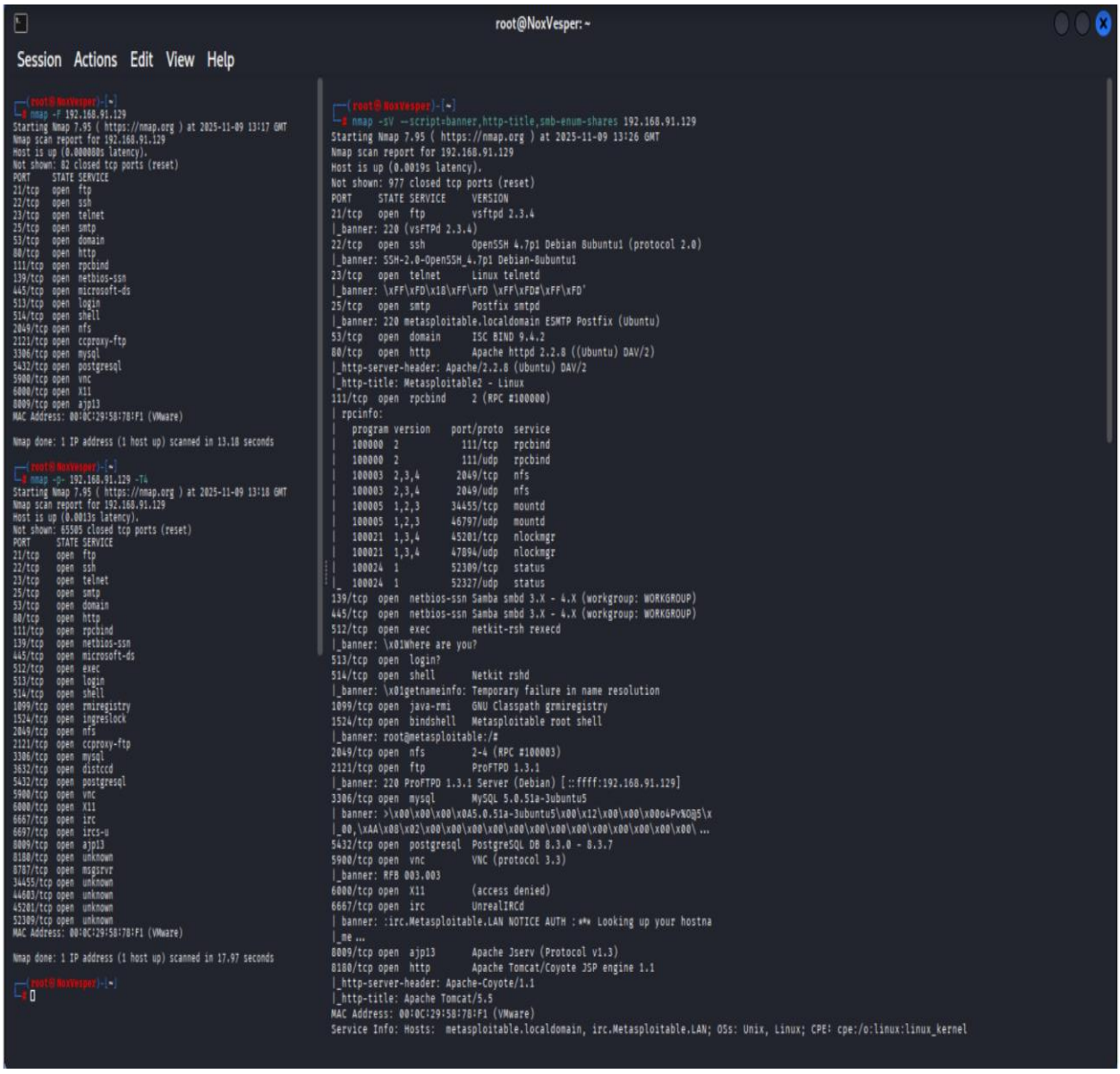
RECON

TASK	PURPOSE	COMMAND FOR KALI	WHAT TO RECORD
Host discovery	Find which hosts / VMs are up on your network	sudo arp-scan -l OR netdiscover -r 192.168.91.0/24	IP, MAC, hostname (if shown), timestamp
Ping sweep & quick banner grab	Verify reachability & grab simple banners	nmap -sn 192.168.91.0/24 then nc -v <IP> <port>	Which Ips respond; simple banner text
Service / version fingerprint	Identify open services & versions for prioritization	nmap -sV -p- --version-intensity 2 <target-ip>	Open ports, service name, version string, confidence



SCANS

SCAN TYPE	WHY USE IT	COMMAND FOR KALI	HOW TO INTERPRET QUICKLY
Quick TCP port scan	Fast way to see common ports open	Nmap F <target-ip>	If many common ports open → investigate services listed
Full TCP port scan	Find all listening TCP ports	Nmap -p- <target-ip> -T4	List all ports; highlight high-value ports (80/443/139/445/3306)
Service enumeration scripts	Get deeper info per service (safe scripts first)	Nmap -sV --script=banner,http-title,smb-enum-shares <target-ip>	Look for readable shares, web titles, exposed info; flag version mismatches



REPORTS:

REPORT SECTION	PURPOSE	WHAT TO INCLUDE	EXAMPLE
Executive summary	High-level findings for quick review	Target IP, top 3 issues, risk level (Low/Med/High)	192.168.91.129 – 3 high-risk services (SMB anonymous, old vsftpd, outdated Tomcat) – HIGH
Technical findings	Detailed evidence for each issue	Port, service, version, command used, raw output snippet, screenshots	Port 21 – vsftpd 2.3.4 – anonymous login allowed – command: ftp; evidence: listing /pub/creds.txt
Recommended next steps	Clear, prioritized actions for follow-up	What to test next (non-destructive), mitigation ideas, resources	Verify SMB share contents (read-only) remove anonymous FTP update Tomcat to X.Y.Z

When I write a professional penetration testing report I divide it into three clear sections:

- Executive Summary
- Technical Findings
- Recommended Next Steps

Each section has a specific audience & purpose, and together they make the report both easy to read.

- In the Executive Summary I give a concise, high-level view of what I found. I state the target IP, the top issues that matter most, and an overall risk rating (Low / Medium / High). This is for managers or anyone who needs a big picture quickly – no technical dumps.
  For example I might write:
  "192.168.91.129 – three high-risk services found (SMB anonymous access, outdated vsftpd, vulnerable Tomcat) – Overall risk: HIGH."
- In Technical Findings I record the evidence & reproduce what I did so system owners can reproduce & fix issues. For every finding I list the port, the service & version, the exact command or tool I used, and include raw output snippets or screenshots where relevant.
  A typical entry looks like:
  "Port 21 – vsftpd 2.3.4. – anonymous login enabled – discovered using ftp; evidence: readable directory /pub/creds.txt (screenshot attached)."
  I keep this section factual & traceable so admins can verify each claim.
- In Recommended Next Steps I give clear, prioritized actions that follow from the findings. I include non-destructive verification steps, short-term mitigations, and longer-term fixes (patches or configuration changes). I make the guidance practical & specific.
  For example:
  "Verify SMB share permissions (read-only where possible), disable anonymous FTP, and update Tomcat to the latest secure release X.Y.Z."
  I also point to resources or compounds admins can use to validate the remediation.

PASSIVE RECONNAISSANCE: WHOIS, NSLOOKUP, DIG, SHODAN OVERVIEW?

WHOIS:

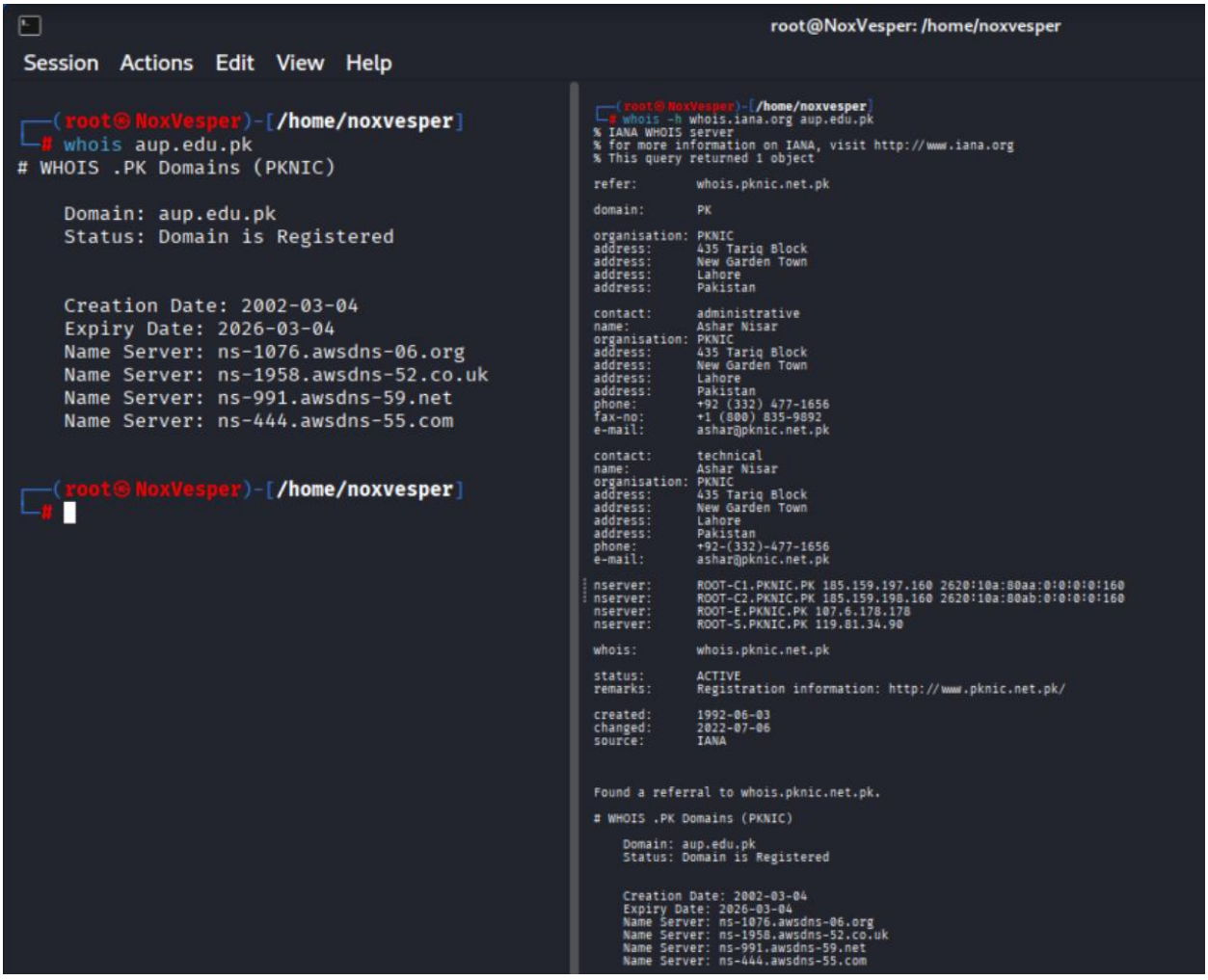
I use WHOIS to gather info about a domain's ownership, registration date, and contact details without touching the target directly.

Example:

- I checked whois aup.edu.pk to find registrar info.
- I used an online WHOIS lookup to get the admin email of a target domain.
- I saw the domain's expiry date to know if it's about the expire.

BASIC COMMANDS FOR WHOIS:

TOOL	COMMAND	RESULT
WHOIS	whois example.com	I query the domain's WHOIS record
	whois -h whois.iana.org example.com	I ask a specific WHOIS server (IANA) for authoritative info



NSLOOKUP:

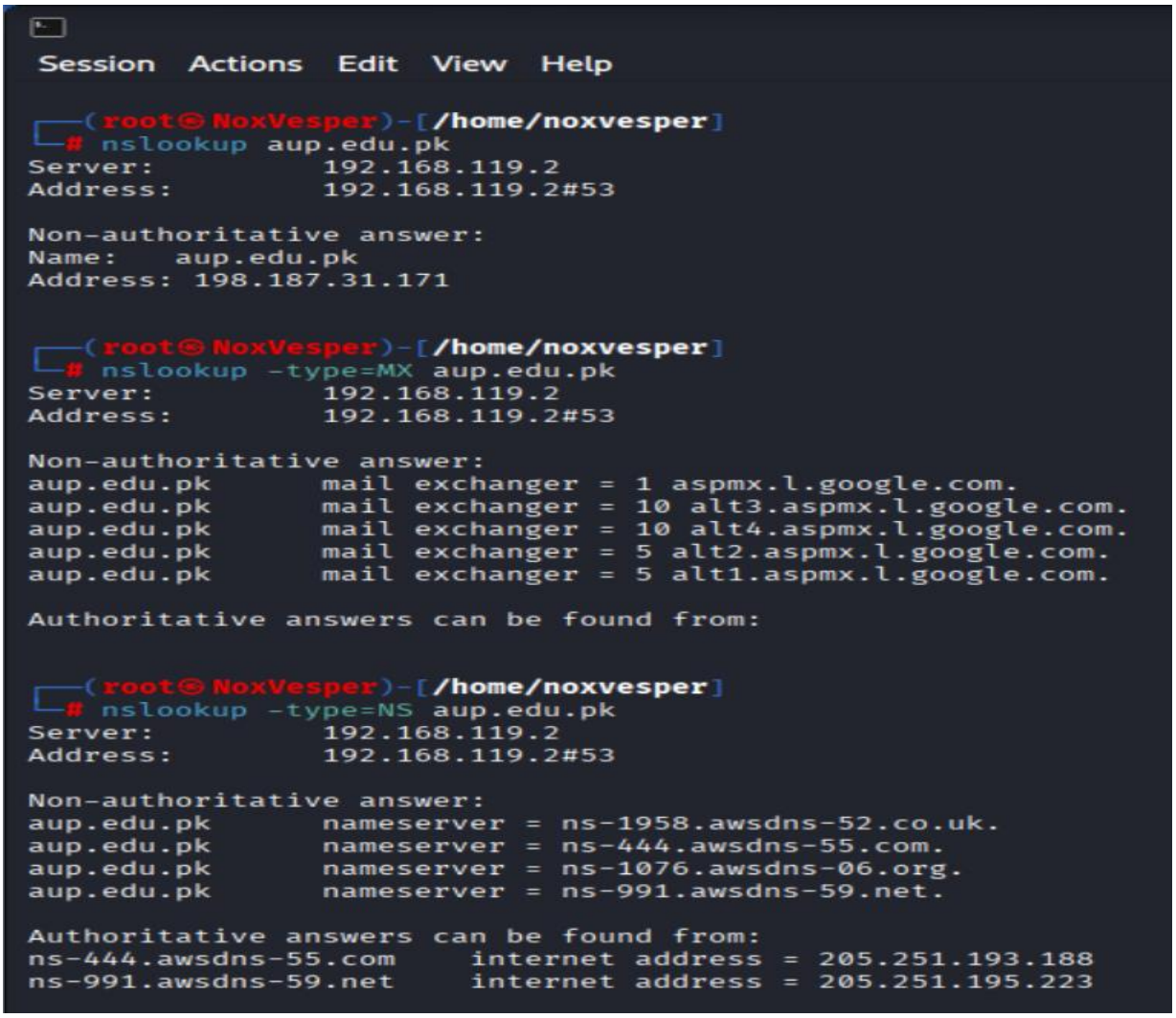
I use nslookup to find a domain's IP address and DNS records. It helps me understand the DNS structure.

Example:

- I ran nslookup facebook.com to get its IP.
- I used nslookup -type=MX yahoo.com to find mail servers
- I found the Name Server (NS) records to check hosting provider

BASIC COMMANDS FOR NSLOOKUP:

TOOL	COMMAND	RESULT
nslookup	nslookup example.com	I resolve the domain to its IP(s)
	nslookup -type=MX example.com	I list the mail (MX) servers for the domain
	nslookup -type=NS example.com	I check the domain's authoritative name servers



DIG:

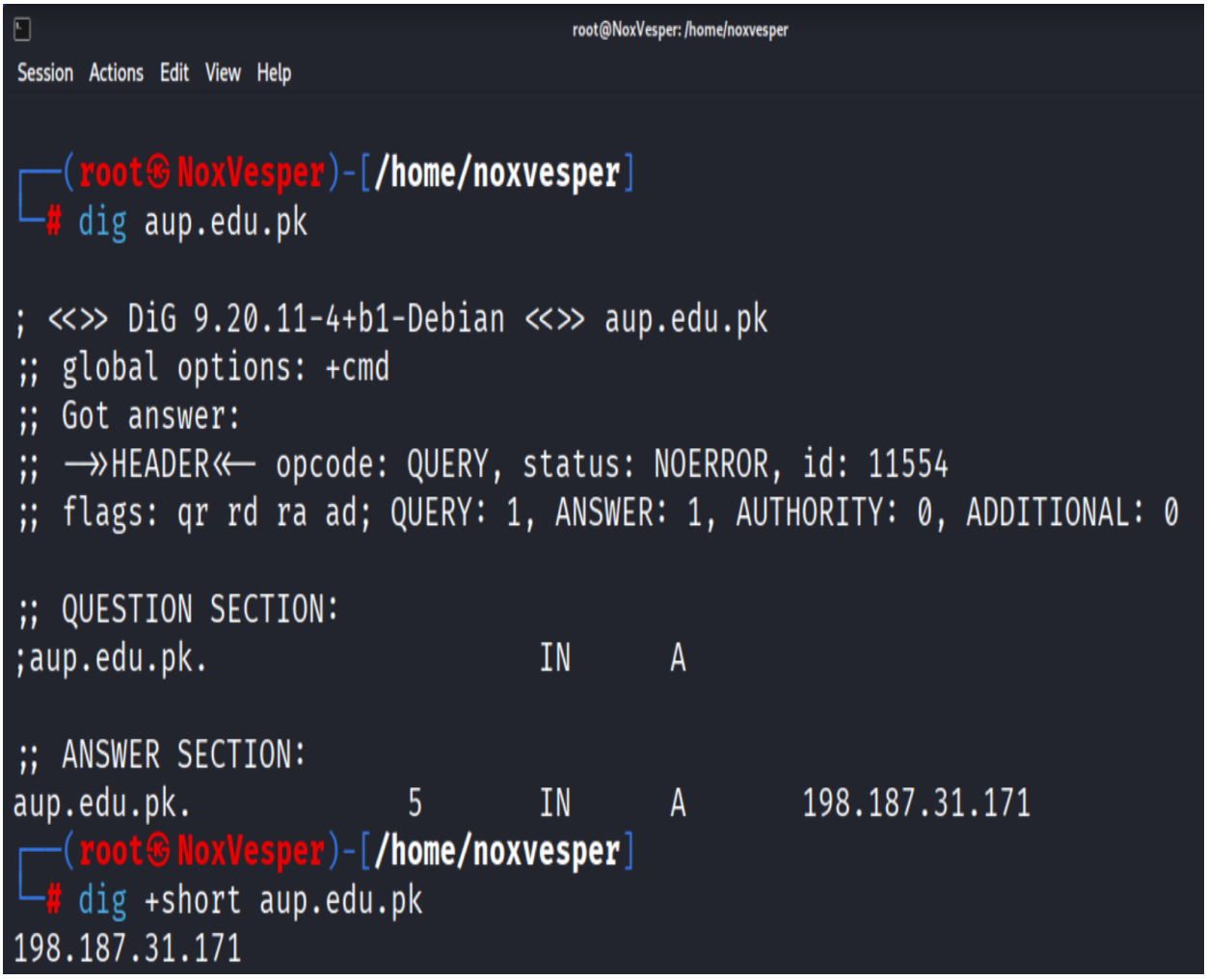
I use dig (Domain Information Groper) for detailed DNS queries & to verify DNS configurations.

Example:

- I used 'dig google.com A' to get IPv4 info.
- I ran 'dig +trace example.com' to follow DNS resolution path.
- I checked 'dig example.com MX' to see mail servers.

BASIC COMMANDS FOR NSLOOKUP:

TOOL	COMMAND	RESULT
Shodan	dig example.com A	I request the domain's A (IPv4) record
	dig +short example.com	I get a concise answer – just the resolved IP(s)
	dig +trace example.com	I trace the DNS



```
┌──(root㉿NoxVesper)-[/home/noxvesper]
└─# dig +trace aup.edu.pk

; <<>> DiG 9.20.11-4+b1-Debian <<>> +trace aup.edu.pk
;; global options: +cmd
.                   5       IN      NS      i.root-servers.net.
.                   5       IN      NS      f.root-servers.net.
.                   5       IN      NS      h.root-servers.net.
.                   5       IN      NS      c.root-servers.net.
.                   5       IN      NS      d.root-servers.net.
.                   5       IN      NS      m.root-servers.net.
.                   5       IN      NS      g.root-servers.net.
.                   5       IN      NS      b.root-servers.net.
.                   5       IN      NS      j.root-servers.net.
.                   5       IN      NS      e.root-servers.net.
.                   5       IN      NS      l.root-servers.net.
.                   5       IN      NS      k.root-servers.net.
.                   5       IN      NS      a.root-servers.net.
.                   5       IN      RRSIG   NS 8 0 518400 20251123050000 20251110040000 61809 . 6jL4QcWAAAAABTLdO3Jk1pLz6R1usptIWhrwQpiHj1g3VhdrCpraptj4
 ZMfnIRTt58FnDTAprh/3HCxWo/8Rmx9gXkJhpUF3bYv5jf/wDR+ywmnf /74csd2Z7kHQBoHCogor+FvBqg4/HoSpPl6mbXCoon9OfU4VKLj+2513 esH5N0j+oowt+xhoh/ncHrerLuY7d1zzsyPlIiilt
KoPHkNk+rC4+Ysv uiBXr/zvE5haKpdPCYVXja2GNWTlBnd5doqyjIM95c1K/hJF02kuNYit H3dYSzQw9XulWmItfvWQ7Ub55MXKa+ivAWT9qrXu1KdwffkXi5r7l1Kz Rfgokw=
;; Received 1125 bytes from 192.168.119.2#53(192.168.119.2) in 112 ms

pk.                 172800  IN      NS      root-e.pknic.pk.
pk.                 172800  IN      NS      root-s.pknic.pk.
pk.                 172800  IN      NS      root-c1.pknic.pk.
pk.                 172800  IN      NS      root-c2.pknic.pk.
pk.                 86400   IN      NSEC    pl. NS RRSIG NSEC
pk.                 86400   IN      RRSIG   NSEC 8 1 86400 20251123050000 20251110040000 61809 . o/IWTisTgrZdUQA6PQ4I/sapnLx58knBq3i8GOoJwMqNT5zl2qU2bem
5 0mfuLfnYignSkUYr+5TseAOjs7vd2t3YP2MlhN95zoGtmMJEun06Ly8D 0DxGVcJ/7RFBb85YBO0EHYGdm7BXdk5bnf3ZwB5mudt5/ihecnxHUem/ Xie05r84yqt95YsqOLGe6gxwePZrLoOWzwlMI8FX
kM7q385Or6FWg09Z jfwGq+qdOmZ8xvGDPwVfo6CjNCcpLxGdJHRpmIZ88Okwq/QT7S9bI5yl XiIuryOCP6xzMclUatdlm8iZhhhOdeVEKrPMm+gbhishI0GizXC1G2qc F9oBQg=
;; Received 562 bytes from 192.58.128.30#53(j.root-servers.net) in 644 ms

aup.edu.pk.         86400   IN      NS      ns-444.awsdns-55.com.
aup.edu.pk.         86400   IN      NS      ns-991.awsdns-59.net.
aup.edu.pk.         86400   IN      NS      ns-1076.awsdns-06.org.
aup.edu.pk.         86400   IN      NS      ns-1958.awsdns-52.co.uk.
;; Received 179 bytes from 185.159.198.160#53(root-c2.pknic.pk) in 624 ms

;; UDP setup with 2600:9000:5304:3400::1#53(2600:9000:5304:3400::1) for aup.edu.pk failed: network unreachable.
;; no servers could be reached
;; UDP setup with 2600:9000:5304:3400::1#53(2600:9000:5304:3400::1) for aup.edu.pk failed: network unreachable.
;; no servers could be reached
;; UDP setup with 2600:9000:5304:3400::1#53(2600:9000:5304:3400::1) for aup.edu.pk failed: network unreachable.
aup.edu.pk.         300     IN      A       198.187.31.171
aup.edu.pk.         172800  IN      NS      ns-1076.awsdns-06.org.
aup.edu.pk.         172800  IN      NS      ns-1958.awsdns-52.co.uk.
aup.edu.pk.         172800  IN      NS      ns-444.awsdns-55.com.
```

SHODAN:

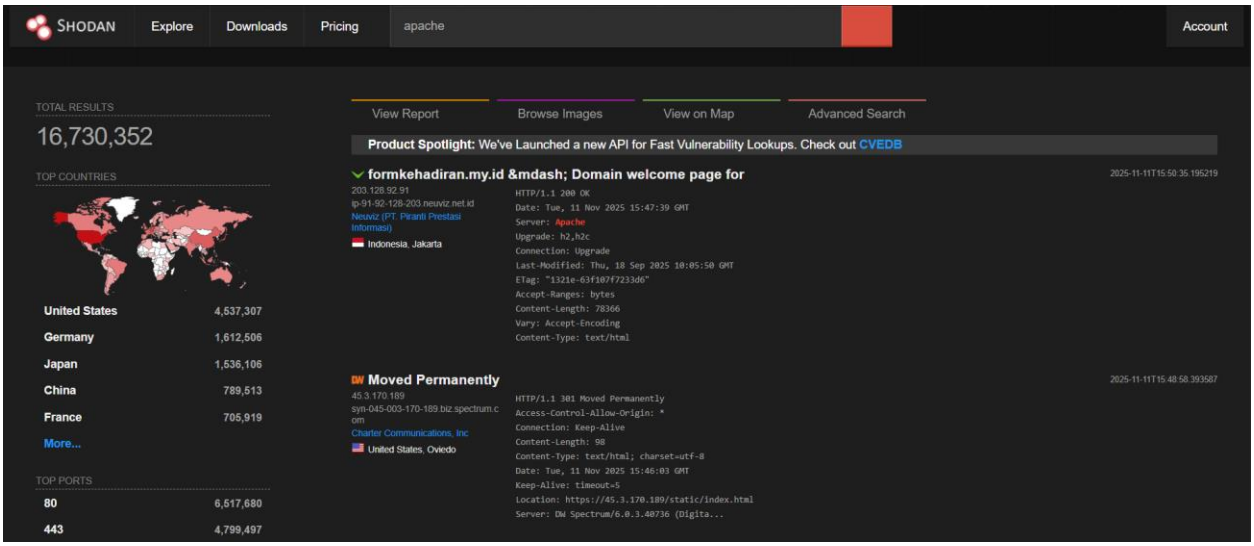
I use Shodan find internet-connected devices and exposed services. It's like a search engine for hackers.

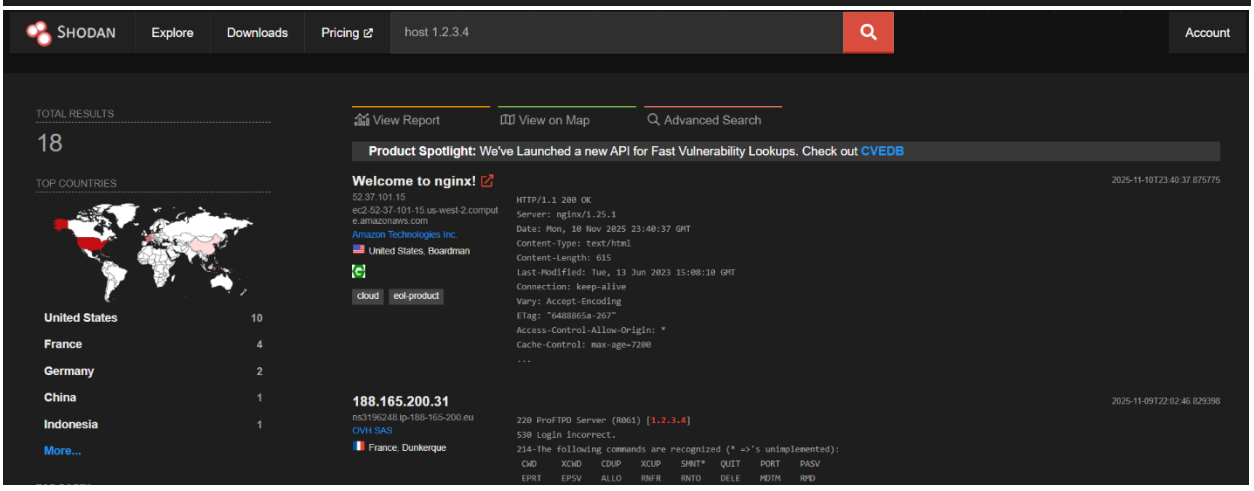
Examples:

- I searched "port:22 country:PK" to find open SSH servers.
- I looked up my own public IP on Shodan to check exposed port.
- I searched for "apache" to see vulnerable web servers.

BASIC COMMANDS FOR SHODAN:

TOOL	COMMAND	RESULT
Shodan	search "apache"	I search Shodan for devices / services mentioning "apache"
	host 1.2.3.4	I inspect a specific IP's Shodan results and open ports





ACTIVE RECONNAISSANCE: NMAP, WHATWEB, WAPPALYZER BASICS?

ACTIVE RECONNAISSANCE:

When I perform active reconnaissance, I directly interact with the target system to collect details like open ports, services, and technologies in use. This helps me understands the target's setup before launching any attack.

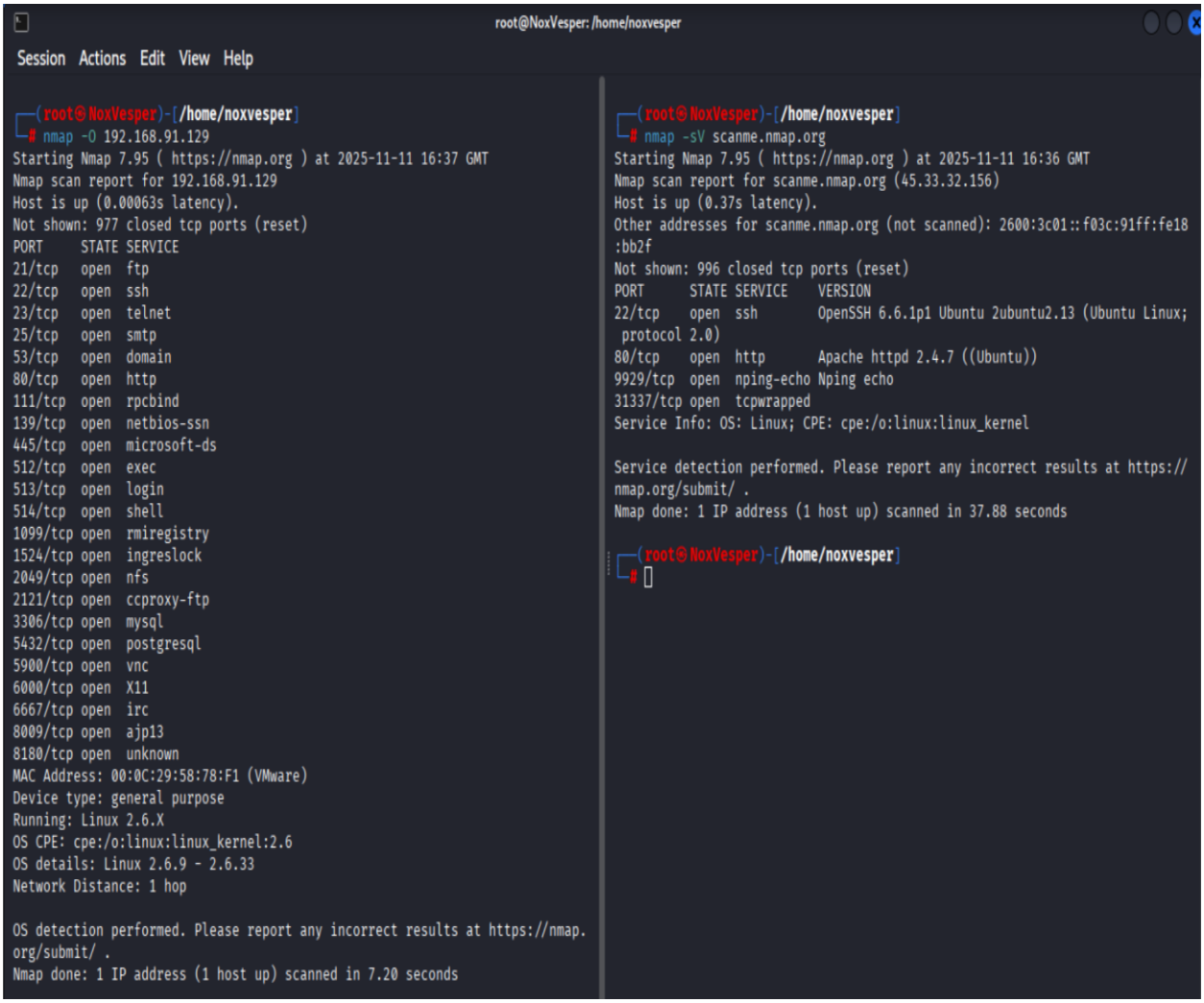
NMAP:

I use Nmap to find open ports, running services, and the operating system of a target.

It helps me know which network services are available for further testing.

BASIC COMMANDS OF NMAP:

COMMAND	PURPOSE	OUTPUT
Nmap -sV example.com	Detect service versions	Reveals services running on ports
Nmap -O example.com	Detect operating system	Detects OS



WHATWEB:

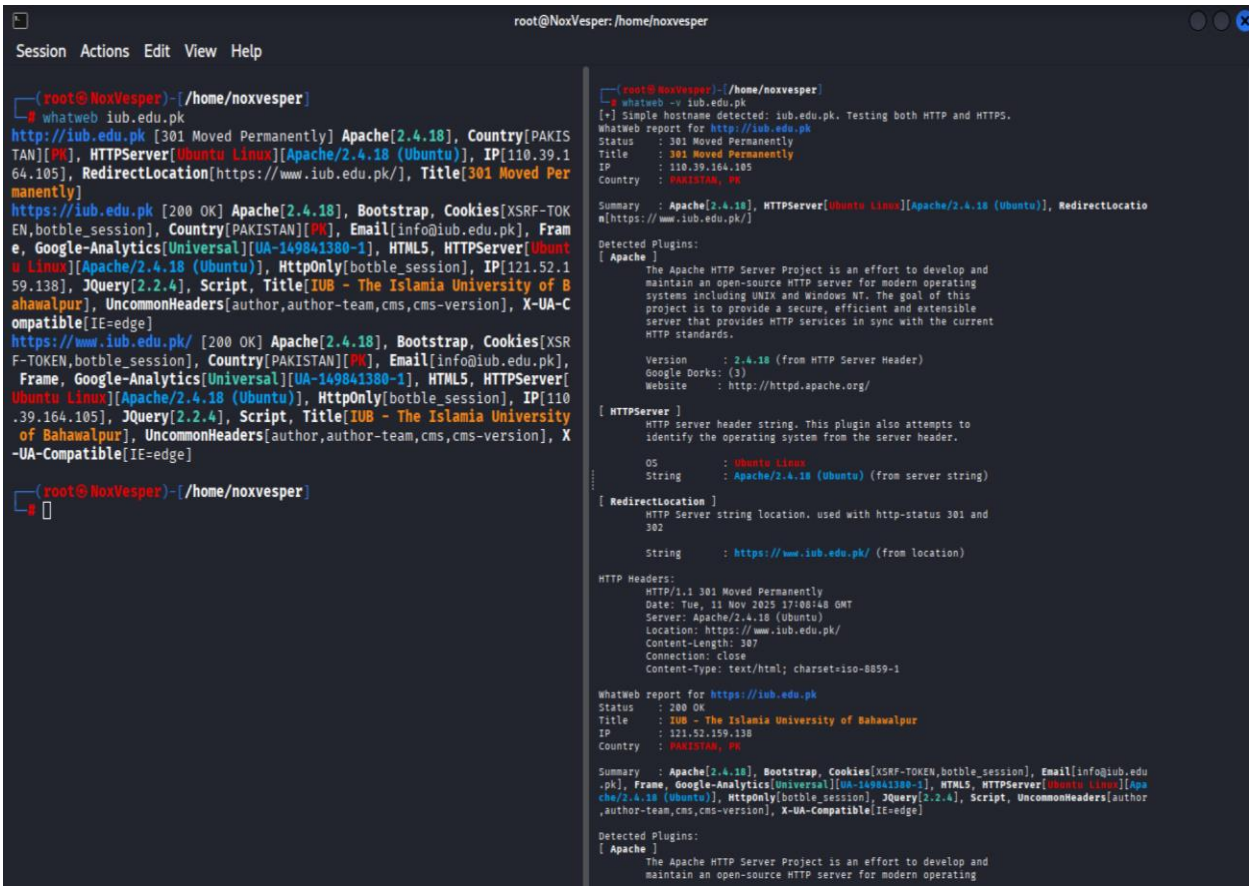
I use WhatWeb to identify what technologies a website is using.

It tells me about the CMS, frameworks, plugins, and web servers behind a website.

- CMS (Content Management System) is a software that helps me create and manage website content easily without needing to code much.
  Example: WordPress, Joomla
- Frameworks is a ready-made structure or base for building web apps faster and more securely.
  Example: Django, React
- Plugins is a small add-ons that extend the features or functions of a website or software.
  Example: SEO plugin for WordPress
- Web Servers are programs that deliver website pages to users when they visit a site.
  Example: Apache, Nginx

BASIC COMMANDS FOR WHATWEB:

COMMAND	PURPOSE	OUTPUT
whatweb example.com	Identify web technologies	Detects WordPress, Apache
whatweb -v example.com	Verbose output for more info	Shows detailed header and CMS version



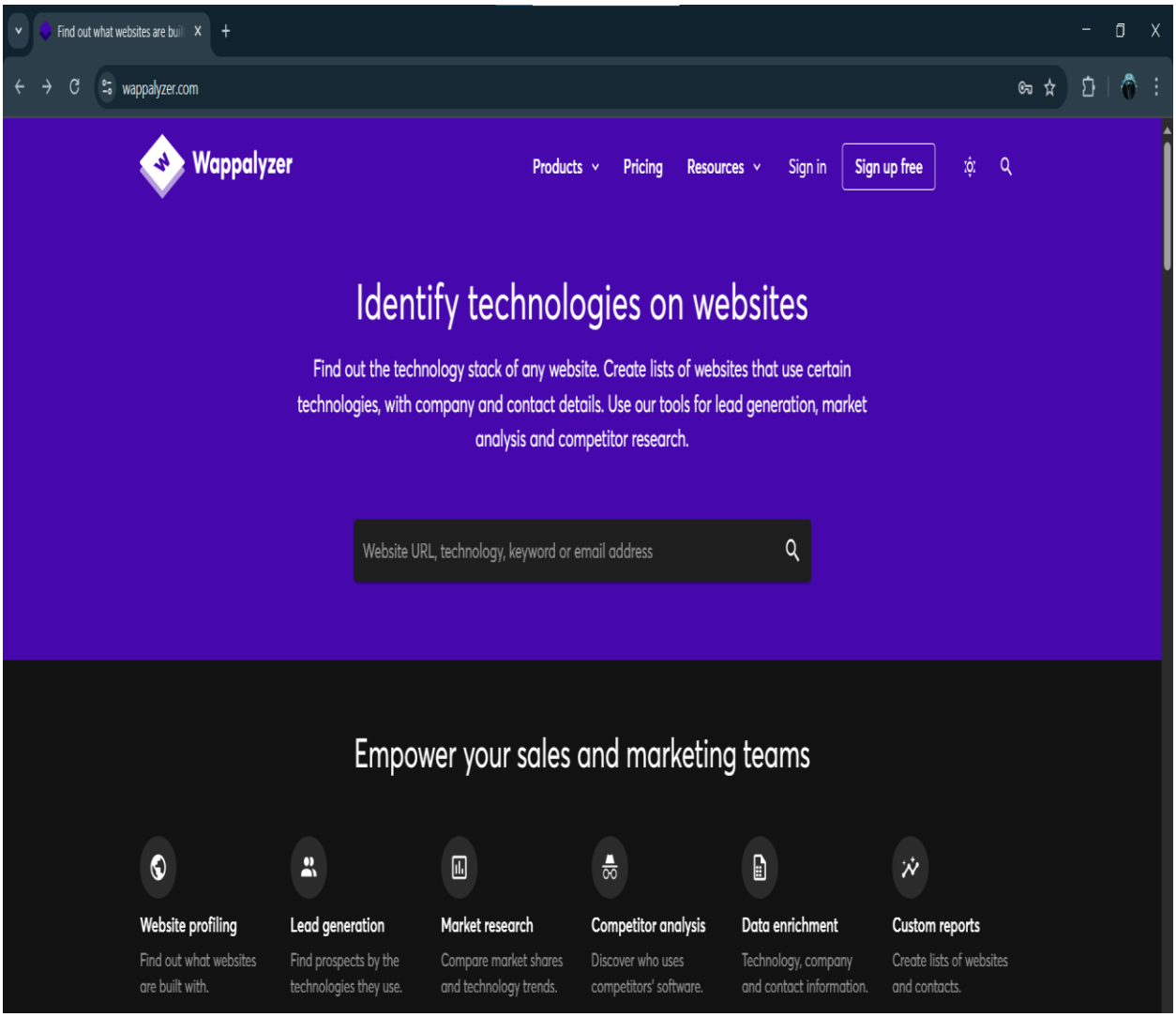
WAPPALYZER:

Wappalyzer is a tool that I use to identify the technologies used on a website. It tells me which CMS, frameworks, programming languages, web servers, and analytics tools a site is built with. It works as a browser extension or command-line tool, helping me quickly understand a site's tech stack.

I use Wappalyzer (browser extension or CLI) to quickly detect technologies used on websites like frameworks, CMS, or databases.

BASIC COMMANDS FOR WAPPALYZER:

COMMAND	PURPOSE	OUTPUT
Visit site with Wappalyzer extension	Auto-detects tech stack	Shows: React, Nginx, Google Analytics
Use CLI: wappalyzer https://example.com	Scan via command line	Detects: Bootstrap, PHP, MySQL
Compare two sites' tech stacks	Understand differences	Site A uses Django, Site B uses WordPress



DIFFERENCE BETWEEN OPEN-SOURCE INTELLIGENCE (OSINT) AND RECON?

When I talk about the difference between OSINT and reconnaissance, I see reconnaissance as the broader process of gathering information about a target, while OSINT is just one part of it.

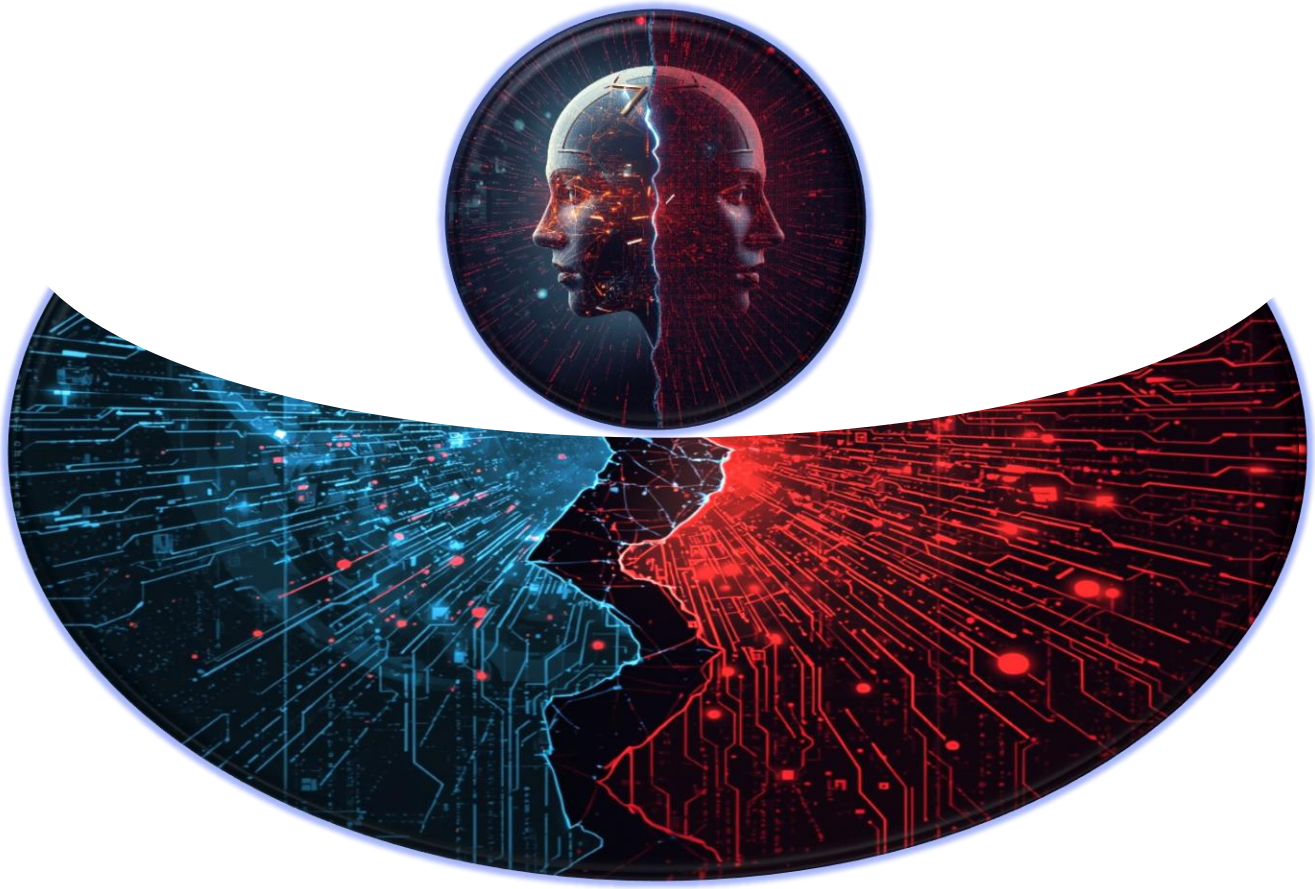
Reconnaissance includes two main types – passive & active.

In passive recon, I collect information without directly touching the target, and this is where OSINT plays its role. For example, if I'm analyzing a company, I might use Google Dorking, LinkedIn, or Whois lookup to find their public emails, employee names, and domain records. This helps me understand the organization's structure quietly, without alerting anyone.

Then comes active recon, where I interact directly with the target's systems to collect deeper technical details. For instance, I might use tools like Nmap to scan open ports or WhatWeb to detect what technologies the company's website is running on.

So, from my perspective, OSINT is like observing a distance, while active reconnaissance is like knocking on the door to see who answers – both are important, but I always start with OSINT to stay stealthy and informed.

So OSINT is a part of passive recon.



CHOOSE A PUBLIC TARGET SUCH AS JUICE-SHOP.HEROKUAPP.COM

PERFORM THE FOLLOWING TASKS:

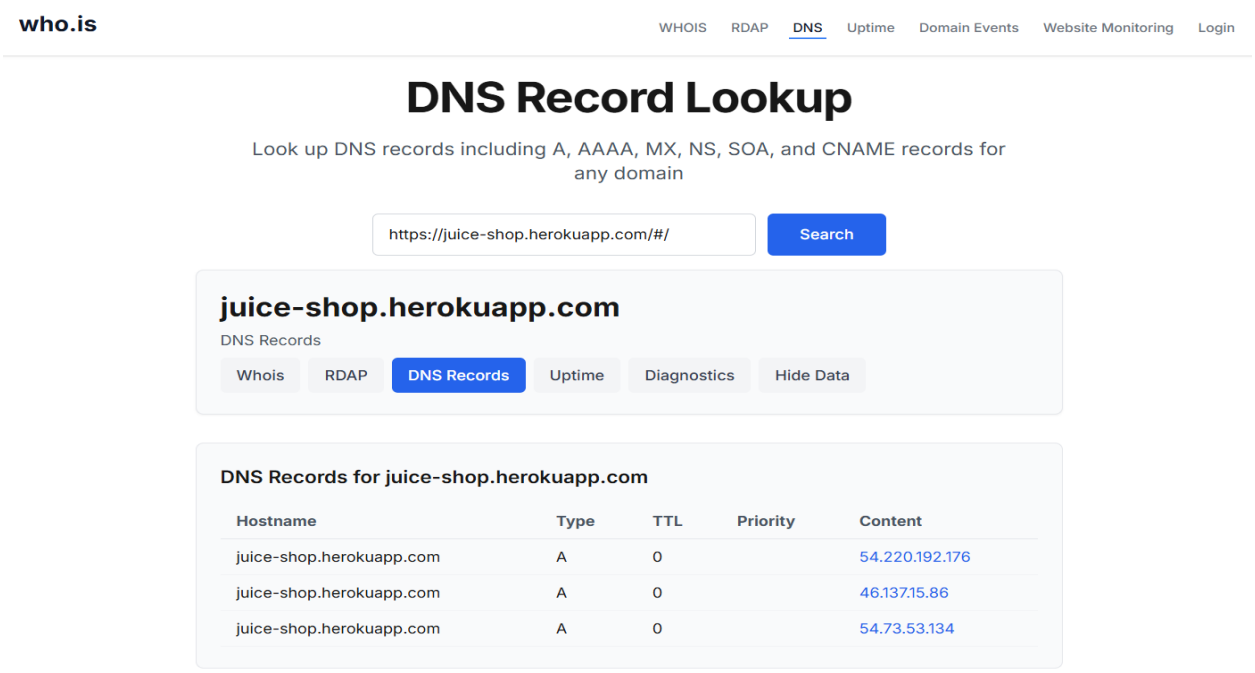
WHOIS LOOKUP (IDENTIFY ORGANIZATION, IP RANGE)

When I talk about **WHOIS lookup**, I like to think of it as a digital identity check for any website on the internet. Whenever I want to know who owns a domain, when it was created, or where it is registered, I use WHOIS as my primary tool. When I perform a WHOIS lookup, I'm basically asking the global domain registry to show me public record of a specific domain name. This record gives me details like the owner's name (if it isn't hidden), the registrar company, contact email, IP addresses, and the expiration and creation date of the domain. I use this information to investigate suspicious websites, verify authenticity, and understand the background of a domain during my cybersecurity work. For me, WHOIS lookup works like checking the ID card of a website simple, informative, and extremely useful for ethical hacking and security analysis.

Juice-shop.herokuapp.com?

This website is actually the OWASP Juice Shop, one of the most famous intentionally vulnerable web applications in the world. Whenever I open Juice Shop, I'm basically entering a practice playground for ethical hacker. The whole purpose of this website is to teach cybersecurity, web hacking, and penetration testing through hands-on challenges.

Juice Shop looks like a normal online store at first – selling juices, smoothies, and products – but behind the scenes, it is packed with vulnerabilities. I can practice SQL injection, XSS, broken authentication attacks, insecure file uploads, IDOR, misconfigurations, and dozens of real-world security flaws. Every weakness I exploit teaches me how attackers think and how defenders should protect applications.



NETWORK SCANNING & SERVIE ENUMERATION:

UNDERSTANDING TCP / IP BASICS:

When I study networking, I see TCP / IP as the backbone of how devices communicate over the internet. TCP / IP stands for Transmission Control Protocol / Internet Protocol. I think of IP as the addressing system that tells data packets where to go, like a postal address, and TCP as the courier that ensures the packets arrive complete and in order.

I understand TCP / IP in layers:

- **Application Layer:**
  Where I use apps like browsers or email (HTTP, FTP, SMTP).
- **Transport Layer:**
  TCP / UDP ensures reliable delivery and error checking.
- **Internet Layer:**
  IP handles addressing and routing.
- **Network Access Layer:**
  Manages physical transmission over cables or Wi-Fi.

NMAP SCANNING TYPES (PING, TCP SYN, SERVICE DETECTION):

When I use Nmap, I rely on different scan types depending on what information I want from a target. When I start with a Ping Scan, I simply check if the host is alive by sending probes; I don't scan ports yet – I just confirm whether the system is reachable. When I perform a TCP SYN Scan (the famous stealth scan), I send only the initial SYN packet to each port. If I receive SYN / ACK, I know the port is open, and I immediately reset the connection without completing the handshake, making my scan fast and harder to detect. For deeper analysis, I use Service Detection, which sends probes to open ports to identify what services and versions are running, such as Apache, OpenSSH, or MySQL.

This helps me understand the software behind each port and what potential vulnerabilities might exist.

How will you ensure safe scanning?

When I perform any kind of scan – whether it's Nmap, WHOIS, or vulnerability testing – I always follow safe scanning practices to avoid legal issues and accidental damage. First, I make sure I have clear permission from the owner of the system; if I don't have authorization, I simply don't scan. Before starting, I test my commands on my own lab environment, so I don't accidentally launch aggressive scans on a real network. During scanning, I begin with non-intrusive methods like ping and port discovery instead of jumping directly to heavy or exploit-based scans. I also monitor the network load, because I don't want my scan to slow down someone's service or trigger unnecessary alarms.

Finally, I document everything to do, so if something goes wrong, I can trace my steps. These safe practices help me stay ethical, avoid detection problems, and keep my penetration testing work professional.make

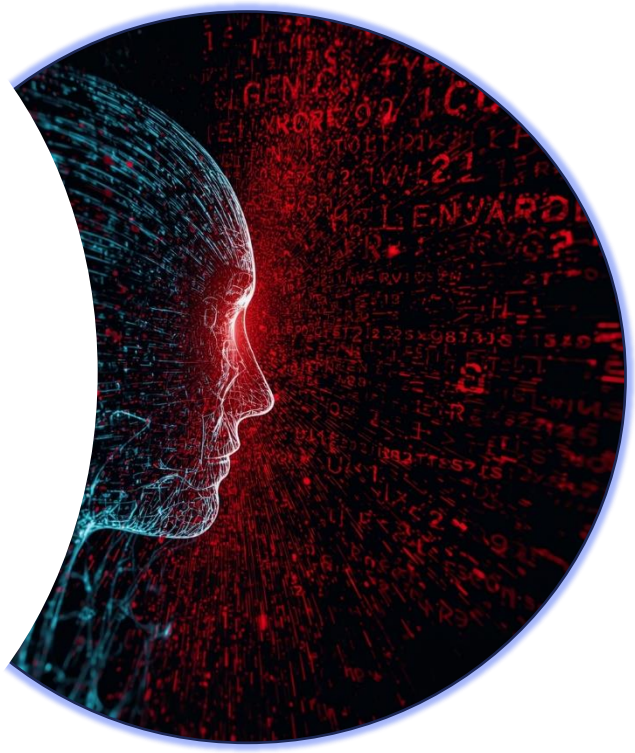
DOCUMENTATION & REPORTING:

WHY REPORTING MATTERS MORE THAN HACKING?

When I work in cybersecurity, I've learned that hacking is only half of the job; the real value comes from how well I report what I discovered. Anyone can find vulnerabilities, but if I can't explain them clearly, the organization can't fix them. A good report turns my technical findings into something actionable, understandable, and useful for decision-makers. It shows the exact risk, the impact, and the steps needed to fix the issue. Reporting also proves my professionalism, protects me legally, and builds trust with clients. In the end, hacking shows my skills, but reporting shows my responsibility – and that is why I consider reporting far more important.

HOW TO WRITE FINDINGS CLEARLY AND PROFESSIONALY?

- **Structure the Report:**
  I organize it with a summary, followed by detailed findings, their impact, and severity.
- **Explain Each Findings:**
  I describe what I found, how I found it, and why it matters in simple, understandable language.
- **Provide Evidence:**
  I include screenshots, logs, or any proof to support each findings.
- **Give Recommendations:**
  I suggest clear, actionable steps to fix or mitigate the issues.
- **Maintain Professional Tone:**
  I suggest clear, actionable steps to fix or mitigate the issues.
- **Proofread:**
  I check the report for clarity, grammar, and accuracy before submission.



The first phase of this research is completed.