Fast & Effective Recon

unvalidor

Fast & Effective Recon v2

Hey!
This little checklist is taken from the book Bug Bounty Bootcamp by Vickie Li and a little bit of my own knowledge. I hope it helps you.

- Ways to contact me :
  - t.me/unvalidor
  - x.com/unvalidor
  - github.com/unvalidor
- Note :
  I tried to make this book concise and fast-paced, and still complete.
  It's possible that I didn't explain how to use certain tools, or that I mentioned a method in the tips without providing an example.
  So search, ask, learn more and enjoy the process.
  Help me improve this book too :)

Dedicated to the one who knows. 🌙

Section 0: Recon Mindset & Prioritization Strategy

Why Mindset Matters?

Recon is not just about running tools — it's about thinking like an attacker, understanding the business, and identifying what truly matters. A strategic mindset helps you focus on high-impact findings and avoid wasting time on noise.

Key Principles

1. **Think Like a User:**
   Explore the application as a regular user would. Understand workflows, permissions, and interactions.
2. **Think Like a Developer:**
   Predict how features are built, where shortcuts might exist, and what might be exposed unintentionally.
3. **Think Like an Attacker:**
   Look for trust boundaries, misconfigurations, and assumptions that can be broken.

Prioritization Framework

Priority	Target Type	Reason
High	Authenticated endpoints	Often under-tested and rich in logic flaws
High	Admin panels	Sensitive functionality and elevated privileges
Medium	Public APIs	May expose data or allow abuse
Medium	Third-party integrations	Often overlooked and misconfigured

Priority	Target Type	Reason
Low	Static content	Less likely to yield vulnerabilities

Business Context Awareness

- What does the company care about most?
- What data is sensitive or valuable?
- What features drive revenue or user engagement?
- What would be most damaging if compromised?

Reporting Strategy

- Focus on impact, not just technical severity
- Include reproduction steps, screenshots, and clear explanations
- Suggest realistic fixes or mitigations
- Be respectful and professional — reputation matters

Section 1: Manual Recon & Scoping

What Is Manual Recon?

Manual reconnaissance is the process of exploring a target application by hand — without automated tools — to understand its structure, user flows, and potential attack surfaces.

Key Activities

- **Browse the Application:** Click through all pages, links, and features.
- **Create Multiple Accounts:** Use different roles (admin, user, guest) to see how the app behaves for each.
- **Identify Input Points:** Look for forms, search bars, upload fields, and query parameters.
- **Observe User Interactions:** How do users interact with each other? Messaging, sharing, commenting?
- **Check for Hidden Features:** Look for unlinked pages, developer tools, or debug endpoints.
- **Understand Business Logic:** What is the app trying to do? Where could logic flaws exist?

Attack Surface Checklist

Area	Examples
URLs	`/admin` , `/dashboard` , `/api/v1/users`
Parameters	`?id=123` , `?role=admin`
Headers	`X-Forwarded-For` , `User-Agent`
Cookies	`sessionid` , `auth_token`
File Uploads	Profile pictures, documents
User Roles	Admin, Moderator, Guest
Third-Party Integrations	Payment gateways, analytics

Tips

- Use browser dev tools to inspect network requests and hidden fields.

- Try accessing unauthorized pages directly via URL manipulation.
- Look for inconsistencies between user roles.
- Take notes and screenshots — they’ll help later during exploitation.

Section 2: Scope Discovery & Asset Enumeration

What Is Scope Discovery?

Scope discovery is the process of identifying all digital assets that belong to a target — including domains, subdomains, IPs, cloud storage, and third-party services. This helps define what you're allowed to test and where to focus your recon efforts.

Asset Types to Discover

Type	Examples
Domains	`example.com`, `example.org`
Subdomains	`api.example.com`, `dev.example.com`
IP Addresses	`192.0.2.1`, `203.0.113.5`
Cloud Buckets	`s3://company-data`, `gs://project-assets`
Mobile Apps	iOS/Android APKs
GitHub Repos	`github.com/company/project`
Third-Party Services	`zendesk.company.com`, `company.atlassian.net`

Tools for Scope Discovery

- **AssetFinder** — Finds domains and subdomains from public sources
  `https://github.com/tomnomnom/assetfinder`
- **Amass** — Powerful asset enumeration tool
  `https://github.com/owasp-amass/amass`
- **Subfinder** — Fast subdomain discovery
  `https://github.com/projectdiscovery/subfinder`
- **Chaos Dataset** — Public bug bounty asset list
  `https://chaos.projectdiscovery.io/`
- **SecurityTrails API** — Historical DNS and domain data
  `https://securitytrails.com/`

Cloud Asset Discovery

- Search for exposed buckets using:
  - `https://buckets.grayhatwarfare.com/`
  - Google Dork: `site:s3.amazonaws.com company_name`

Tips

- Always check the program’s official scope (on HackerOne, Bugcrowd, etc.).
- Use multiple tools and merge results for completeness.
- Look for wildcard DNS entries (`*.example.com`) — they often hide dynamic subdomains.
- Monitor DNS changes over time using tools like `dnsdumpster`, `SecurityTrails`, or `PassiveTotal`.

Section 3: Whois & IP Intelligence

What Is Whois?

Whois is a protocol used to query databases that store information about domain registrations. It helps identify who owns a domain, when it was registered, and where it points.

Basic Whois Lookup

```
whois example.com
```

Key Fields to Note:

- `Registrant Name`
- `Registrar`
- `Creation Date`
- `Expiration Date`
- `Name Servers`
- `Registrant Email`

Reverse Whois

Find all domains registered with the same email, company name, or address.

Tools:

- ViewDNS Reverse Whois
- Whoxy
- SecurityTrails

What Is IP Intelligence?

IP intelligence involves gathering information about IP addresses associated with your target — including their hosting provider, ASN, geolocation, and other domains sharing the same IP.

IP Lookup & Reverse IP

```
nslookup example.com
```

Reverse IP Lookup Tools:

- ViewDNS Reverse IP
- YouGetSignal
- Censys

ASN & IP Range Discovery

```
whois <ip_address>
```

Look for the `NetRange` and `OrgName` fields.

Advanced ASN Mapping:

```
whois -h whois.cymru.com "157.242.2.20"
```

Use this to identify Autonomous System Numbers (ASNs) and map related IP blocks.

Tips

- Reverse IP can reveal staging or forgotten domains hosted on the same server.
- Whois data may be redacted — use historical tools like `SecurityTrails` or `DomainTools`.
- Monitor IP changes over time to detect infrastructure shifts.
- Combine IP intelligence with subdomain enumeration for deeper asset discovery.

Section 4: Certificate Analysis & Host Discovery

Why Analyze SSL Certificates?

SSL/TLS certificates often contain valuable metadata that can help you discover additional hosts, subdomains, and services related to your target. This is especially useful for identifying assets that aren't publicly linked or indexed.

Key Field: Subject Alternative Name (SAN)

The SAN field lists all domains and subdomains covered by the certificate. These can include:

- Internal subdomains
- Development environments
- Third-party integrations

Tools for Certificate Discovery

Tool	Description
crt.sh	Search public certificate transparency logs
Censys	Search and analyze certificates and hosts
Cert Spotter	Certificate monitoring and discovery
Google Certificate Transparency	View certificate logs from Google

Example Query (crt.sh)

```
https://crt.sh/?q=example.com
```

Use `output=json` to parse results programmatically.

Automation Tools

- **CertSh Fetcher:**
  `https://github.com/nahamsec/crtsh` — Automates crt.sh queries
- **CertStream:**
  Real-time certificate stream for live monitoring
  `https://github.com/CaliDog/certstream-python`

Tips

- Look for wildcard certs (`*.example.com`) — they often imply dynamic subdomain generation.
- Monitor newly issued certs to catch fresh assets.

- Combine certificate data with subdomain enumeration tools for deeper coverage.
- Use historical certificate data to find deprecated or forgotten services.

Section 5: Subdomain Enumeration

Why Subdomains Matter?

Subdomains often host development environments, APIs, admin panels, and forgotten services. They expand the attack surface and are frequently overlooked by defenders.

Enumeration Techniques

Passive Enumeration

- Uses public data sources without interacting with the target.
- Tools:
  - Subfinder
  - AssetFinder
  - Amass
  - SecurityTrails
  - Censys
  - crt.sh

Active Enumeration

- Sends DNS queries or probes to discover subdomains.
- Tools:
  - DNSx
  - Gobuster
  - MassDNS

Wordlists

- Use high-quality wordlists from:
  - SecLists
  - Commonspeak2
  - Custom lists based on target's tech stack or naming conventions

Sample Gobuster Command

```
gobuster dns -d example.com -w wordlist.txt -t 50
```

- `-d` : Target domain
- `-w` : Wordlist
- `-t` : Threads

Tips

- Always resolve discovered subdomains to check if they're live.
- Test for sub-subdomains (e.g., `dev.api.example.com`).
- Use wildcard DNS detection to avoid false positives.
- Combine passive and active methods for full coverage.
- Monitor for new subdomains over time — recon is never one-and-done.

Section 6: Service Enumeration & Port Scanning

Why Service Enumeration Matters?

Once you’ve discovered domains and subdomains, the next step is to identify what services are running on them. Open ports and exposed services can reveal web servers, databases, APIs, and other entry points for exploitation.

Active Scanning

Use tools that directly probe the target to identify open ports and running services.

Nmap

```
nmap -sV -Pn -T4 example.com
```

- `-sV` : Detect service versions
- `-Pn` : Skip host discovery (useful for firewalled targets)
- `-T4` : Faster execution

RustScan (Fast wrapper for Nmap)

```
rustscan -a example.com -- -sV
```

Passive Intelligence

Use third-party databases to discover services without touching the target.

Tool	Description
Shodan	Search engine for internet-connected devices
Censys	Certificate and host discovery
ZoomEye	Chinese alternative to Shodan
Project Sonar	Internet-wide scan data

Common Ports to Watch

Port	Service
80 / 443	HTTP / HTTPS
21	FTP
22	SSH
25 / 587	SMTP
3306	MySQL
5432	PostgreSQL
6379	Redis
27017	MongoDB

Tips

- Scan both domain names and IP addresses.
- Use stealthier scan options (`-sS`, `-T2`) for sensitive targets.
- Combine results from active and passive methods.
- Look for unusual ports — they often hide custom services.
- Save scan results in formats like XML or JSON for later parsing.

Section 7: Spidering & Crawling

What Is Spidering?

Spidering is the process of automatically crawling a website to discover all accessible URLs, endpoints, forms, and resources. It helps uncover hidden paths, dynamic content, and areas that manual browsing might miss.

GUI-Based Crawlers

Burp Suite (Professional Edition)

Steps:

1. Configure browser to use Burp as a proxy
2. Log in manually to capture session cookies
3. Right-click target domain in the Target tab → "Spider this host"
4. Set scope to limit crawling to the target domain
5. Enable form submission to discover deeper content
6. Review results in Target → Site Map

Export URLs:

- Right-click → "Copy URLs" or "Save as CSV"

OWASP ZAP

Usage:

1. Launch ZAP and set the target URL
2. Use Spider for static crawling
3. Use AJAX Spider for JavaScript-heavy sites
4. Review results under the Sites tree

Command-line:

```
zap-cli spider https://example.com
zap-cli ajax-spider https://example.com
```

CLI-Based Crawlers

Hakrawler

```
echo https://example.com | hakrawler -depth 3 -subs -js -robots -sitemap -threads 10
```

Options:

- `-depth`: Crawl depth

- `-subs`: Include subdomains
- `-js`: Parse JavaScript files
- `-robots`, `-sitemap`: Include robots.txt and sitemap.xml

Photon

```
python3 photon.py -u https://example.com -t 3 -o output/
```

- `-t`: Threads
- `-o`: Output directory

LinkFinder

```
python3 linkfinder.py -i https://example.com/app.js -o cli
```

Purpose: Extract endpoints from JavaScript files

Tips

- Always crawl after authentication to access protected areas
- Use session cookies or headers in CLI tools to mimic logged-in users
- Export discovered URLs for brute forcing, fuzzing, or scanning
- Combine spidering with JavaScript analysis to uncover hidden APIs
- Respect robots.txt but never rely on it to hide sensitive paths

---

Section 8: Hidden Parameter Discovery & JavaScript File Analysis

Why It Matters?

Hidden parameters and JavaScript files often expose undocumented functionality, internal APIs, and sensitive logic. These are prime targets for IDORs, privilege escalation, and logic flaws.

Hidden Parameter Discovery

Practical Manual Techniques

- **Use Burp Suite (or browser dev tools):**
  - Browse the app while intercepting requests
  - Look for parameters like `user_id`, `admin`, `debug`, `env`, `feature_flag`, `redirect`, `next`, etc.
- **Modify and Replay Requests:**
  - Add parameters manually in Burp: `?isAdmin=true`, `?role=admin`, `?debug=1`
  - Observe response changes (status codes, content, redirects)
- **Check HTML and JS Source:**
  - Look for hidden `<input>` fields or commented-out parameters
  - Search for `name="..."` and `value="..."` pairs
- **Try Common Parameter Names:**
  Use lists like:
  - `admin`, `access`, `auth`, `token`, `user`, `role`, `redirect`, `next`, `debug`, `feature`, `env`, `config`

Fast Automated Tools

```
# ParamSpider
python3 paramspider.py -d target.com

# Arjun
python3 arjun.py -u https://target.com/api -m GET

# Kiterunner (for route + param brute-force)
kr scan target.com -w routes.txt -p params.txt
```

Use these tools to find parameters across endpoints, including those not visible in the UI.

JavaScript File Analysis

What to Look For?

- API endpoints like `/api/user`, `/admin/config`, `/internal/feature`
- Tokens, keys, secrets (e.g., `Bearer`, `JWT`, `apiKey`)
- Feature flags or debug toggles
- Hidden routes or undocumented functions

Practical Manual Techniques

- **Download JS files:**
  Use `wget`, browser dev tools, or `getJS`
- **Search for keywords:**
  ```
  grep -Ei 'token|auth|api|key|debug|config|secret' *.js
  ```
- **Use JS beautifiers:**
  If code is minified, use tools like jsbeautifier.org or `prettier`
- **Trace logic:**
  Follow function calls to see how parameters are used or validated

Automated Tools in Action

```
# LinkFinder
python3 linkfinder.py -i https://target.com/app.js -o cli

# SecretFinder
python3 SecretFinder.py -i https://target.com/main.js -o cli

# SubJS + GetJS
subjs -i domains.txt
getJS --url https://target.com
```

These tools extract endpoints, secrets, and parameter names from JS files quickly.

Tips

- Analyze JS after login — many APIs are only exposed to authenticated users
- Use discovered endpoints in `ffuf`, `nuclei`, or manual fuzzing
- Watch for feature flags like `beta=true`, `debug=1`, `showHidden=true`
- JS often reveals internal logic that backend doesn't validate properly
- Combine JS analysis with Burp's Param Miner for deeper coverage

---

Section 9: Directory & File Brute Forcing

Why It Matters?

Web applications often contain hidden directories, backup files, configuration files, and endpoints that aren't linked anywhere but are still accessible. Brute forcing helps uncover these forgotten or sensitive resources.

Tools for Directory Discovery

FFUF (Fast Web Fuzzer)

```
ffuf -u https://example.com/FUZZ -w /path/to/wordlist.txt -t 50 -mc 200,403
```

- `FUZZ`: Placeholder for brute-forced words
- `-mc`: Match status codes (e.g., 200 OK, 403 Forbidden)

Dirsearch

```
python3 dirsearch.py -u https://example.com -e php,txt,log -w /path/to/wordlist.txt
```

- `-e`: File extensions to try

Gobuster

```
gobuster dir -u https://example.com -w /path/to/wordlist.txt -x php,txt,log -t 50 -s 200,403
```

- `-x`: Extensions
- `-s`: Status codes to show

Screenshot Tools

- **EyeWitness**

```
EyeWitness -f urls.txt --web --no-prompt
```

- **Snapper**

```
snapper -i urls.txt -o screenshots/
```

Wordlists

- Use curated lists from `SecLists`:
  - `/Discovery/Web-Content/`
  - `/Fuzzing/`
- Try extensions like `.bak`, `.zip`, `.old`, `.conf`, `.env`, `.inc`

Tips

- Combine brute forcing with spidering and Google Dorking to prioritize likely endpoints.
- Use proxy tools (e.g., Burp Suite) to monitor and replay requests.
- Respect rate limits — adjust thread count and add delays if needed.

- Look for status codes like 403 (Forbidden) and 401 (Unauthorized) — they often indicate something valuable.
- Export discovered URLs for further fuzzing or vulnerability scanning.

Section 10: Cloud Storage & Bucket Recon

Why It Matters?

Many companies use cloud storage services like Amazon S3, Google Cloud Storage, and Azure Blob Storage to host files. Misconfigured buckets can expose sensitive data such as credentials, source code, or internal documents.

Manual Discovery Techniques

Google Dorking

```
site:s3.amazonaws.com company_name
site:amazonaws.com "confidential"
```

Use variations to search for exposed buckets or files.

Bucket Search Engines

- GrayHatWarfare
- LeakIX
- BuckHacker (if available)

Automated Tools

- `lazys3`
- `bucket-stream`
- `S3Scanner`

These tools brute-force bucket names or monitor for public access.

Interacting with Buckets (AWS CLI)

```
# Install AWS CLI
pip install awscli

# List contents
aws s3 ls s3://bucket_name/

# Download file
aws s3 cp s3://bucket_name/file.txt .

# Upload file
aws s3 cp file.txt s3://bucket_name/

# Delete file
aws s3 rm s3://bucket_name/file.txt
```

Tips

- Look for `.env`, `.bak`, `.zip`, and `.log` files

- Check bucket permissions: public read/write access is a red flag
- Monitor for newly created buckets using certificate transparency or GitHub leaks
- Use region-specific endpoints to bypass restrictions

Section 11: GitHub Recon & Code Intelligence

Why It Matters?

GitHub is a goldmine for sensitive information. Developers often unintentionally expose secrets, credentials, internal endpoints, and configuration files in public repositories. Recon on GitHub can reveal assets that are not discoverable through traditional methods.

Manual Techniques

- Search for company name, product name, or domain in GitHub’s search bar
- Explore repositories, issues, pull requests, and commit history
- Check developer profiles linked to the company
- Look for `.env`, `config.php`, `settings.py`, or `.gitignore` files
- Review commit diffs for removed secrets or hardcoded tokens

Automated Tools

Tool	Purpose
`GitRob`	Scans public repos for sensitive files
`truffleHog`	Finds high-entropy secrets in Git history
`Gitleaks`	Detects secrets and credentials in Git repos
`repo-supervisor`	Monitors repos for sensitive content
`GitHub Search CLI`	Automates GitHub search queries

GitHub Dorking Examples

```
company_name password
company_name filename:.env
company_name filename:config.json
company_name api_key
```

Use advanced search operators like `filename:`, `extension:`, and `path:` to narrow results.

Tips

- Monitor GitHub over time — secrets may be exposed in new commits
- Use GitHub’s API for automation and large-scale scanning
- Check forks and clones — sensitive data may persist even after deletion
- Look for internal tools, staging domains, and undocumented APIs
- Combine GitHub recon with subdomain enumeration and certificate analysis

---

Section 12: Google Dorking & Search Engine Recon

What Is Google Dorking?

Google Dorking is the art of using advanced search operators to uncover sensitive information, hidden pages, and misconfigured files indexed by search engines.

Common Dorks

Dork	Purpose
`site:example.com`	Search only within a specific domain
`inurl:"admin"`	Find URLs containing "admin"
`intitle:"index of"`	Discover open directories
`filetype:pdf`	Locate PDF documents
`"password"`	Search for exposed credentials
`ext:log`	Find log files
`site:s3.amazonaws.com company_name`	Discover exposed cloud buckets

Advanced Dorking Techniques

- **Subdomain Discovery:**
  `site:*.example.com`
- **Bucket Hunting:**
  `site:amazonaws.com "company_name"`
- **Exclude Keywords:**
  `"how to port scan" -nmap`
- **Regex-style Searches:**
  `(sqli | "sql injection")`
- **Platform-Specific:**
  `"how to hack" (reddit.com | facebook.com)`

Tools to Automate Dorking

- `GoogleDorkScanner`
- `GHDB`
- `DorkSearch`

Tips

- Use quotes (`"..."`) for exact matches.
- Combine multiple operators for precision.
- Try different file extensions: `.bak`, `.conf`, `.env`, `.zip`
- Use Google Alerts to monitor new indexed content.
- Respect robots.txt — but don't rely on it to hide sensitive files.

Section 13: OSINT & Public Intelligence Gathering

What Is OSINT?

Open-Source Intelligence (OSINT) refers to the collection and analysis of publicly available data to gain insights about a target. This includes social media, archived websites, paste sites, public documents, and more.

Key OSINT Sources

Source	Description
Wayback Machine	View historical versions of websites
URLScan.io	Analyze URLs and their behavior
Hunter.io	Find email addresses associated with domains
ViewDNS	DNS, Whois, IP, and reverse lookup tools
Pastebin	Search for leaked credentials or internal data
PublicWWW	Search source code across the web
LinkedIn	Identify employees, roles, and technologies used
Google Groups	Discover internal discussions or exposed threads

OSINT Tools

- `PasteHunter` — Monitors paste sites for sensitive data
- `theHarvester` — Collects emails, subdomains, and names
- `SpiderFoot` — Automated OSINT scanner
- `Recon-ng` — Modular OSINT framework
- `FOCA` — Extracts metadata from public documents

Endpoint Discovery from Archives

Waybackurls

```
cat domains.txt | waybackurls > archived_urls.txt
```

Purpose: Fetch all URLs saved by the Wayback Machine for a domain

Tips

- Use archived endpoints for fuzzing and vulnerability scanning
- Monitor paste sites for leaked credentials or tokens
- Cross-reference employee names with GitHub and LinkedIn
- Look for internal tools, staging environments, and forgotten assets
- Automate OSINT workflows with tools like SpiderFoot or Recon-ng

---

Section 14: Technology Stack Fingerprinting

Why It Matters?

Understanding the technologies behind a web application helps identify known vulnerabilities, misconfigurations, and potential attack vectors. Fingerprinting reveals frameworks, CMS platforms, server software, and third-party libraries in use.

Manual Techniques

- **View Page Source:**
  Look for comments, meta tags, and headers like `X-Powered-By`, `Server`, or `Generator`

- **Check HTTP Headers:**
  Use browser dev tools or tools like `curl` and `httpx`

```
curl -I https://example.com
```

- **Look for JavaScript Libraries:**
  Identify frameworks like React, Angular, jQuery, etc.
- **Search for Error Messages:**
  Stack traces or debug output can reveal backend technologies

Automated Tools

Tool	Purpose
Wappalyzer	Browser extension for tech profiling
BuiltWith	Online tool for stack analysis
`WhatWeb`	CLI tool for fingerprinting
`WebTech`	Detects technologies and versions
`nmap -sV`	Detects service versions on open ports

Vulnerability Databases

- CVE Details
- MITRE CVE Search
- Exploit Database
- Vulners

Use these to search for known vulnerabilities in identified technologies.

Tips

- Combine fingerprinting with subdomain and service enumeration for deeper insights
- Pay attention to outdated versions — they often have public exploits
- Look for default installations or exposed admin panels
- Use multiple tools to cross-verify results
- Monitor tech stack changes over time — updates may introduce new weaknesses

---

Section 15: Recon Automation & Workflow Optimization

Why Automate Recon?

Manual recon is powerful, but automation helps scale your efforts, reduce repetition, and ensure consistency across targets. A well-structured recon pipeline saves time and uncovers more assets.

Key Benefits

- Faster asset discovery
- Repeatable workflows
- Easier integration with bug bounty platforms
- Continuous monitoring of targets

Automation Frameworks

Tool	Description
ReconFTW	Full automation of recon process
Nahamsec’s Recon Profile	Curated tools and scripts
ProjectDiscovery Suite	Includes Subfinder, HTTPx, Nuclei, etc.
LazyRecon	Bash script for quick recon
OneForAll	Subdomain enumeration framework

Sample Workflow

1. **Subdomain Enumeration:**
   `Subfinder`, `Amass`, `AssetFinder`
2. **DNS Resolution:**
   `dnsx`, `massdns`
3. **Port Scanning:**
   `nmap`, `rustscan`
4. **HTTP Probing:**
   `httpx`, `httprobe`
5. **Screenshotting:**
   `EyeWitness`, `Aquatone`
6. **Directory Brute Forcing:**
   `ffuf`, `gobuster`
7. **Vulnerability Scanning:**
   `Nuclei`, `Nikto`

Scheduling & Monitoring

- Use `cron` jobs or CI/CD pipelines to run recon periodically
- Store results in structured formats (JSON, CSV)
- Use dashboards or alerting systems to track changes

Tips

- Customize wordlists and templates for each target
- Use tagging and filtering to prioritize high-value findings
- Integrate with Slack, Discord, or email for alerts
- Always validate automated findings manually before reporting

---

good luck ✌🏻