SpiderFoot — Professional Cheat Sheet

One-page professional reference for **SpiderFoot** — an automated OSINT reconnaissance and attack surface discovery tool. Practical commands, modules overview, scanning modes, automation/API, integration tips, and OPSEC notes for red-team and threat intelligence workflows.

---

1) At-a-glance

- **Tool:** SpiderFoot (https://www.spiderfoot.net) — automated reconnaissance engine for domains, IPs, names, e-mail addresses, ASNs and more. Collects from 200+ data sources (whois, DNS, web, social, breach DBs, IP geolocation, passive DNS, SSL certs, etc.).
- **Primary uses:** Attack surface discovery, external asset inventory, exposure assessment, threat intel gathering, and SOC enrichment.
- **Deployments:** Local CLI, Docker, or server mode with web UI/API.

---

2) Install / Start quickly

```
# Docker (recommended quick start)
docker run -d --name spiderfoot -p 5001:5001 spiderfoot/spiderfoot:latest
# then open http://localhost:5001

# From pip / source
pip3 install spiderfoot
# run
sf.py -l 0.0.0.0:5001   # start web UI on port 5001
# CLI usage
sf.py -h
```

---

3) Architecture & key components

- **Scan engine**: runs modules to collect data and build correlation graph.
- **Modules**: modular probes (OSINT sources) and classifiers. Enable/disable per scan.
- **Web UI / API**: create/manage scans, view graph, export reports.
- **Data store**: SQLite by default; can configure PostgreSQL for larger deployments.

---

4) When to use which interface

- **Web UI** — best for interactive exploration, graphs, and export.
- **CLI** — good for automation, scripting, and batch scans.
- **API** — integrate with CI, SOAR, or custom dashboards (JSON results, report export).

---

5) Quick CLI examples

```
# Simple scan (domain) via CLI
sf.py -s SF_TEST -t domain -v --target example.com

# Scan an IP address
sf.py -s ip_scan -t ip -v --target 8.8.8.8

# Use config file with module selection
sf.py -s scoped -c ./spiderfoot.conf --target example.com

# Run headless (no UI) and export JSON
sf.py -s batch1 -t domain --target example.com --json output.json
```

Use `sf.py --help` and `sf.py --list-modules` to explore module names and options.

---

6) Important module families (high value)

- **Passive DNS / DNS**: `bing_search`, `virustotal.vt`, `passive_dns` — find historical records and subdomains.
- **WHOIS & domain**: whois, registrar, nameservers, domain age, DNSSEC.
- **SSL / certificate**: `ssl`, `crtsh` — certificate transparency and SANs for subdomain discovery.
- **Web content / discovery**: `robots`, `sitemap`, `pages`, `waybackmachine` — crawl and index content.
- **Breach / credentials**: `haveibeenpwned`, `breachdata` — check for exposed emails/passwords.
- **Social media / identity**: `twitter`, `linkedin` (limited by API), `username` probes.
- **Geolocation / IP owner**: `geoip`, `asnlookup`, `shodan` — map infrastructure and provider.
- **Shodan/Censys/VT**: enrich with internet-wide scanners and reputation feeds.

---

7) Scopes & module selection (best practice)

- **Start narrow:** enable passive and safe modules first (WHOIS, DNS, crtsh, passive DNS, public search).
- **Add enriched sources:** VT, Shodan, Censys, Org/ASN lookups.

- **Add intrusive or authenticated modules only with permission** (services that use credentials or send authenticated queries).
- **Use API keys:** configure keys for paid/enriched sources (Shodan, VirusTotal, Censys, HaveIBeenPwned) to improve results.

---

8) API & automation

- **REST API**: use it to create scans, poll status, and fetch results. Typical flow: `POST /scan` -> `GET /scan/<id>` -> `GET /scan/<id>/export?format=json`.
- **Example (curl)**

```
# create scan
curl -X POST "http://localhost:5001/api/scan" -H "Content-Type: application/json" -d '{"target":"example.com","modules":"all_passive"}'
# export JSON
curl "http://localhost:5001/scan/<id>/export?format=json" -o sf.json
```

- **Automation ideas:** integrate into CI (on deploy), SOAR enrichment, or nightly asset discovery jobs.

---

9) Reporting & exports

- **Formats supported:** HTML, CSV, JSON, PDF (via web UI).
- **Graph view:** interactive graph for pivoting between entities — useful for quick triage.
- **Export tips:** export JSON for programmatic ingestion; HTML for human reports with graphs.

---

10) Performance & scaling

- **Default DB (SQLite)** is fine for single/scoped scans. For enterprise, configure PostgreSQL.
- **Parallelism:** adjust module concurrency in config for faster scans; watch API rate limits for external providers.
- **Caching:** enable cached results for repeated sources (reduces API usage and speedups).

---

11) Integration & enrichment

- **Chain with other tools:** feed subdomains to `gobuster` / `ffuf`, nmap, or burp for deeper testing.
- **Enrich with paid feeds:** Shodan, Censys, VirusTotal, RiskIQ, PassiveTotal for higher fidelity.
- **Connect to Kibana/Elasticsearch**: export JSON and index results for historical tracking and dashboards.

---

12) Common operational examples

- **Asset discovery for new domain:** WHOIS + crtsh + passive DNS + sitemap + wayback -> enumerate subdomains and hosts.
- **Breach exposure check:** HaveIBeenPwned + paste sites + breachdata modules on email address lists.
- **External attack surface mapping:** Shodan + open ports + SSL certs + geolocation -> prioritize high-risk hosts.

---

13) Troubleshooting & pitfalls

- **Missing results:** ensure API keys are set and modules enabled. Some sources throttle or block automated queries.
- **False positives:** validate sensitive findings manually (e.g., exposed files).
- **Rate limits & costs:** paid APIs may bill per query; cache and schedule queries.
- **Dependency issues:** when running from source, ensure required Python packages and tldextract data are up to date.

---

14) OPSEC & legal

- OSINT data collection can touch third-party services and privacy boundaries — verify your engagement scope.
- Avoid sending credentials or destructive queries.
- Coordinate with stakeholders before scanning large orgs or performing repeated automated queries against public APIs.

---

15) One-line cheats & quick runs

```
# Quick domain passive scan via CLI (JSON output)
sf.py -s quick1 -t domain --target example.com --json quick1.json

# Start web UI (local)
sf.py -l 127.0.0.1:5001

# Create scan via API (curl minimal)
curl -X POST "http://localhost:5001/api/scan" -H "Content-Type: application/
json" -d '{"target":"example.com","modules":"all_passive"}'

# Export results to HTML via API (after scan completes)
curl "http://localhost:5001/scan/<id>/export?format=html" -o report.html
```

---

16) Further reading & resources

- Official docs & module list: https://www.spiderfoot.net/documentation/
- SpiderFoot GitHub: https://github.com/smicallef/spiderfoot
- API docs (built into web UI) and examples shipped with the repository.

This cheat sheet is for professional reconnaissance and security testing. Only scan assets you own or are authorized to test.