<div align="center"> ```
```

# LX-BOT ULTIMATE v5.0

**Next-Generation Enterprise Penetration Testing & Bug Bounty Automation Framework**

Show Image

Show Image

Show Image

Show Image

Show Image

Show Image
```

> **Built for professional red teamers, enterprise security teams, and advanced bug bounty hunters.** LX-BOT automates the complete offensive security lifecycle — from passive recon to active exploitation — generating executive-grade reports with CVSS v3.1 scoring.

⚠️ Legal Disclaimer

> **This tool is designed exclusively for authorized security testing.** Only use LX-BOT against systems you have explicit written permission to test. Unauthorized access to computer systems is illegal and unethical. The authors accept no liability for misuse of this software.

🚀 Quick Start

```bash
# 1. Clone the repository
git clone https://github.com/yourusername/lx-bot.git
cd lx-bot

# 2. Install Python dependencies
pip install -r requirements.txt --break-system-packages

# 3. Install all 60+ offensive tools
python3 resource_manager.py --install

# 4. Run your first scan
python3 lx-bot.py -t https://target.com
```

✨ Features

<table> <tr> <td width="50%">

🎯 Complete Attack Surface Coverage

- **10 comprehensive phases** from passive recon to advanced exploitation

- **60+ integrated tools** — no manual setup, no missed angles
- **Full async execution** — parallel tool runs for maximum speed
- **Thread-safe semaphore control** — configurable concurrency

🔍 Intelligent Detection

- **Auto-CMS detection** — WordPress, Joomla, Drupal, Magento, Shopify, Typo3
- **WAF fingerprinting** — 200+ WAF signatures with bypass strategies
- **Service correlation** — Nmap output → searchsploit → Metasploit modules
- **CVE auto-mapping** — Nuclei findings → Exploit-DB → MSF modules

</td> <td width="50%">

📊 Professional Reporting

- **CVSS v3.1** calculated for every finding
- **Executive summary** — board-level risk narrative
- **Interactive Plotly graphs** — severity distribution, CVSS histograms
- **OWASP Top 10 2021** compliance mapping
- **CWE classification** for all vulnerability types
- **Remediation roadmap** — prioritised action plans with timelines

🔗 Perfect Integration

- **Burp Suite proxy** — ALL tool traffic interceptable
- **Metasploit Framework** — automated module search & exploit check
- **report_generator.py** — auto-generates HTML report at scan end
- **resource_manager.py** — one-command tool installation

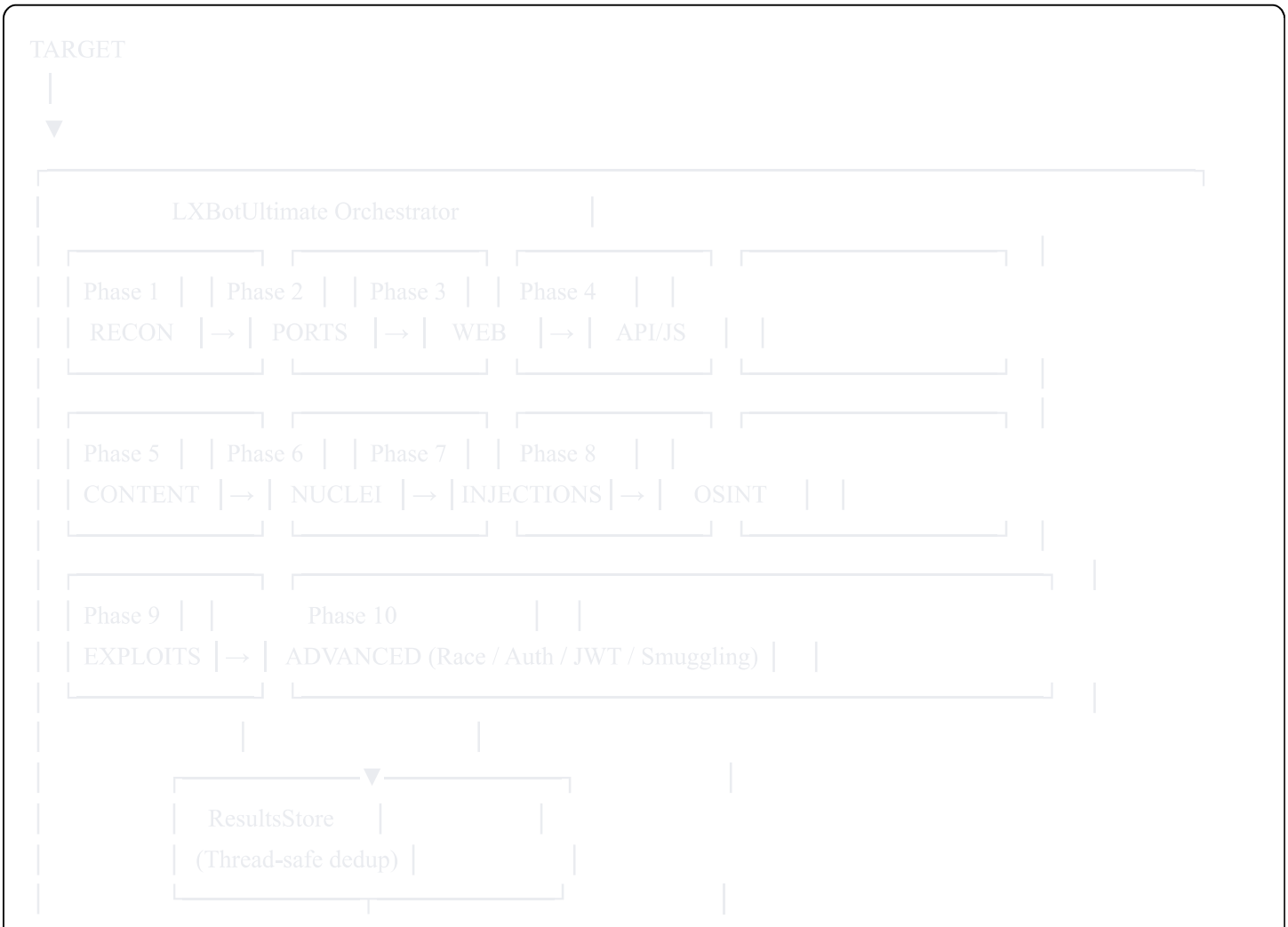
</td> </tr> </table>

📐 Architecture

```
lx-bot/
├── lx.py                ← Core Engine | Phases 1–5 | Base Infrastructure
│   ├── AsyncRunner      | Thread-safe async subprocess executor
│   ├── ResultsStore     | MD5-dedup vuln store, CVSS-aware, thread-safe
│   ├── Phase1Recon      | Subdomain enum, DNS, HTTP probe, screenshots
```

```
│   ├── Phase2Ports          │ Port scanning, service fingerprinting
│   ├── Phase3Web            │ Web analysis, WAF, CMS detection & scanning
│   ├── Phase4API            │ Crawling, JS secrets, API/GraphQL discovery
│   └── Phase5Content        │ Directory fuzzing, git exposure, backup files
│
├── lx-bot.py                ← Orchestrator | Phases 6–10 | CLI Entry Point
│   ├── MetasploitInteg.     │ msfconsole, searchsploit, msfvenom
│   ├── Phase6Nuclei         │ Template-based scanning (5 parallel runs)
│   ├── Phase7Injections     │ XSS, SQLi, LFI, SSTI, SSRF, XXE, CMDi, IDOR
│   ├── Phase8OSINT          │ Email, GitHub secrets, S3 buckets
│   ├── Phase9Exploits       │ CVE correlation, MSF module mapping
│   ├── Phase10Advanced      │ Race conditions, auth bypass, JWT, smuggling
│   └── LXBotUltimate        │ Main orchestrator, report generation, CLI
│
├── resource_manager.py      ← 60+ Tool Installer & Manager
├── report_generator.py      ← Professional HTML Report Generator (CVSS v3.1)
├── requirements.txt         ← Python dependencies
└── README.md                ← This file
```

Data Flow



🛠️ Integrated Tools (60+)

Exploitation Frameworks

Tool	Purpose	Category
msfconsole	Metasploit Framework — Complete exploitation platform	exploitation
msfvenom	Payload generator (reverse shells, stagers, encoders)	exploitation
searchsploit	Offline Exploit-DB search engine	exploitation

CMS Scanners

Tool	Purpose	CMS
wpscan	WordPress vulnerability scanner (plugins, themes, users)	WordPress
joomscan	Joomla component/extension scanner	Joomla
droopescan	Drupal, SilverStripe & Moodle scanner	Drupal
magescan	Magento security assessment	Magento

Network & Port Scanning

Tool	Purpose	Speed
nmap	Deep scan: -sV -sC -O, 300+ NSE scripts	⭐⭐⭐
masscan	Internet-scale port scanner (10M pkts/sec)	⭐⭐⭐⭐⭐
rustscan	Modern Rust-based fast scanner with nmap integration	⭐⭐⭐⭐⭐

Subdomain Enumeration

Tool	Source	Passive/Active
subfinder	40+ passive sources (Shodan, VirusTotal, crtsh…)	Passive
assetfinder	Asset & subdomain discovery	Passive
amass	OWASP in-depth DNS enumeration + brute force	Both
findomain	Multi-source fast enumerator	Passive
chaos	ProjectDiscovery DNS dataset	Passive
subjack	Subdomain takeover detection	Active
subover	Additional takeover fingerprinting	Active

HTTP Analysis

Tool	Purpose
httpx	Fast HTTP probe with tech detection, status, title
whatweb	Technology identification (300+ signatures)
wafw00f	WAF fingerprinting (200+ WAF signatures)
nikto	Web server vulnerability scanner
webanalyze	Deep technology analysis using Wappalyzer rules

Web Crawlers

Tool	Purpose
katana	Next-gen headless crawler with JS execution
gospider	Fast recursive spider with sitemap/robots support
hakrawler	Simple, fast Golang web crawler

JavaScript Analysis

Tool	Purpose
subjs	JS URL extractor from in-scope pages
getJS	JavaScript file discovery & extraction

Fuzzing & Discovery

Tool	Strength
ffuf	Ultra-fast fuzzer, WAF-aware rate limiting, JSON output
feroxbuster	Recursive content discovery with auto-tune
dirsearch	Directory and file search with threading
gobuster	DNS & directory brute force

Vulnerability Scanners

Tool	Purpose
nuclei	Template-based scanner (8,000+ community templates)
jaeles	Signature-based active scanner

Injection Testing

Tool	Targets
sqlmap	SQL injection (12 techniques, 5 databases)
dalfox	Advanced XSS scanner with blind XSS support
commix	Command injection exploitation framework
nosqlmap	MongoDB, CouchDB, Redis NoSQL injection
xsstrike	XSS with DOM analysis and fuzzing

SSL/TLS

Tool	Purpose
testssl.sh	Comprehensive TLS configuration audit
sslscan	SSL/TLS protocol & cipher vulnerability scanner

Screenshots

Tool	Purpose
gowitness	Fast web screenshot utility (Chrome headless)
aquatone	Visual inspection of HTTP surfaces

OSINT

Tool	Purpose
theHarvester	Email, subdomain, name harvesting from 7+ sources
trufflehog	Secret scanning across Git, S3, filesystem
gitleaks	Git repository secret detection
gitrob	GitHub organisation reconnaissance

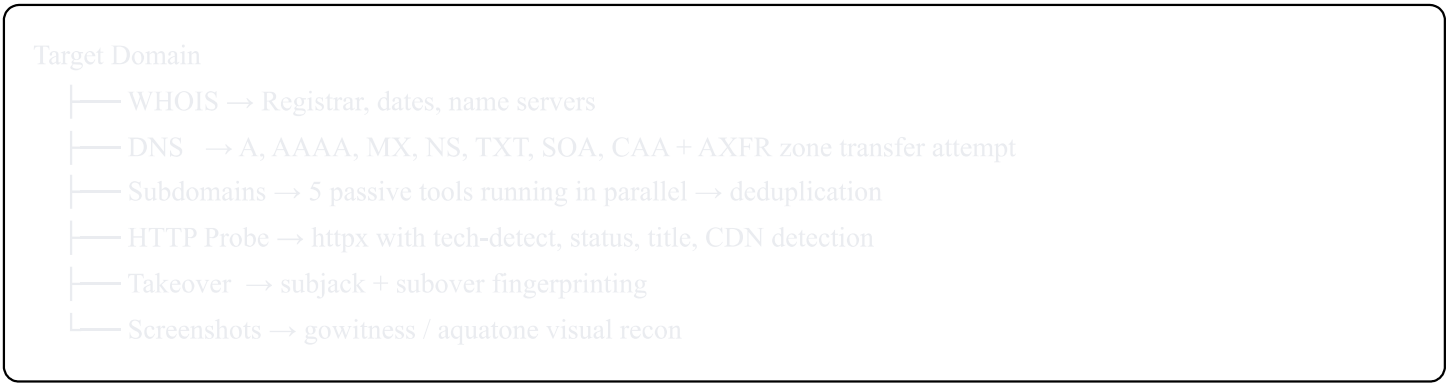
Additional Tools (20+)

`gau` `waybackurls` `gf` `hakrawler` `httprobe` `jq` `git-dumper` `chaos-client` `unzip` `wget` `curl` `whois` `dig` `nslookup` `arjun` `paramspider` `kiterunner` `interactsh-client` `notify` `anew`

📋 10-Phase Methodology

Phase 1 — Reconnaissance 🔍

Tools: `whois` `dig` `subfinder` `assetfinder` `amass` `findomain` `chaos` `httpx` `subjack` `subover` `gowitness` `aquatone`



Findings: DNS Zone Transfer, Subdomain Takeover, Technology Stack

---

Phase 2 — Port Scanning ⚡

Tools: rustscan masscan nmap

Findings: Exposed databases, unencrypted services, risky protocols (Telnet/FTP)

Phase 3 — Web Analysis & CMS Detection 🌐

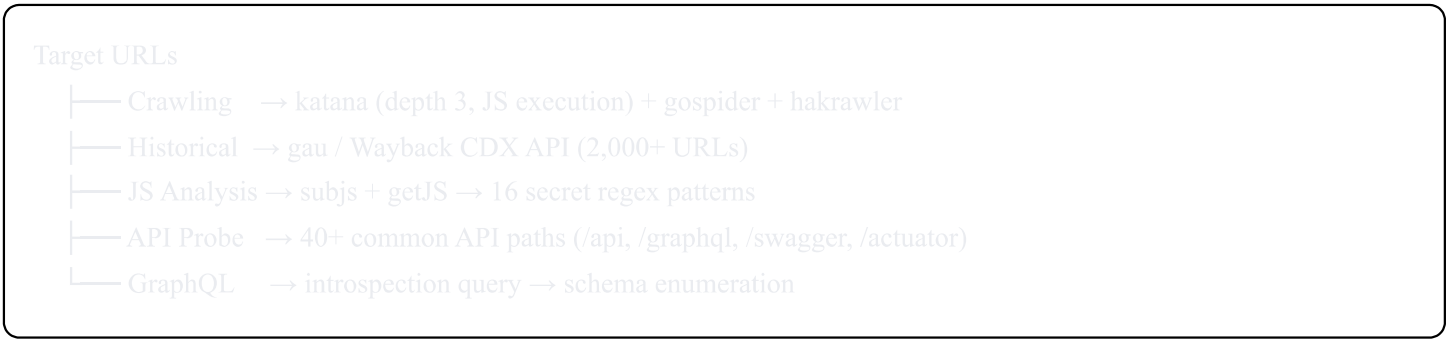
Tools: whatweb wafw00f webanalyze nikto testssl.sh sslscan wpscan joomscan droopescan magescan

Findings: Missing security headers, CORS misconfiguration, weak TLS, CMS vulnerabilities

---

Phase 4 — API & Endpoint Discovery 🔌

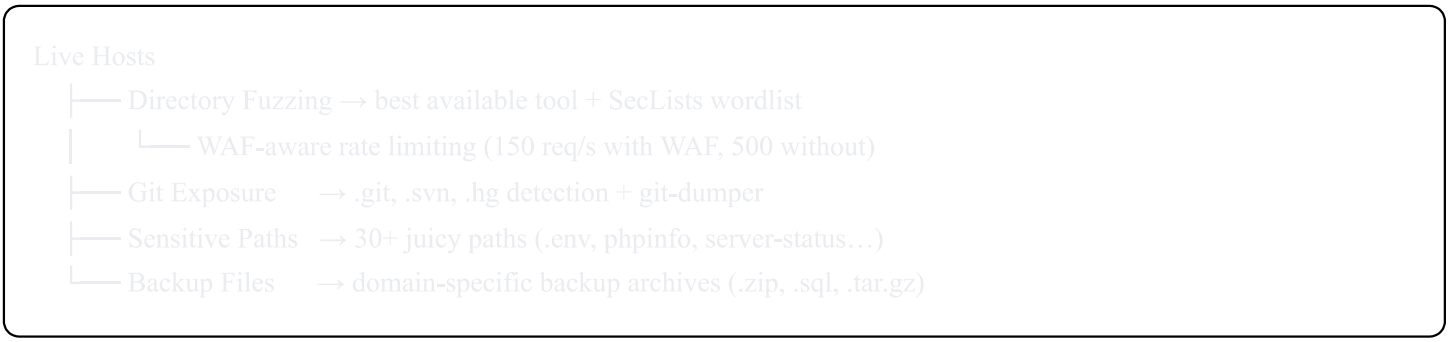
Tools: katana gospider hakrawler subjs getJS gau



Findings: Exposed API keys, GraphQL introspection, undocumented endpoints, secrets in JS

Phase 5 — Content Discovery 📂

Tools: ffuf feroxbuster dirsearch gobuster



Findings: Source code exposure, backup files, admin panels, configuration files

---

Phase 6 — Nuclei Vulnerability Scanning 🎯

Tools: nuclei (5 parallel runs) jaeles

Findings: CVEs with CVSS scoring, exposed admin panels, configuration issues

---

Phase 7 — Injection Testing 💉

Tools: sqlmap dalfox xsstrike commix nosqlmap + manual probes

Vulnerability	Tool	Technique
XSS	dalfox / xsstrike	DOM, reflected, stored, blind
SQL Injection	sqlmap	Boolean, time-based, error, union, stacked
NoSQL Injection	nosqlmap	MongoDB, CouchDB query manipulation
OS Command Injection	commix	Injection, semi-blind, blind
LFI / Path Traversal	manual	php://filter, null byte, traversal chains
SSTI	manual	Jinja2, Twig, FreeMarker, ERB, Pebble
SSRF	manual	AWS metadata, GCP metadata, localhost, gopher
XXE	manual	File read, OOB, SSRF via XXE
Open Redirect	manual	Host header, path, parameter injection
IDOR	manual	Sequential ID delta analysis

Phase 8 — OSINT & Intelligence 🕵️

Tools: theHarvester trufflehog gitleaks gitrob

```
Domain Intelligence
  ├── Email Harvesting  → theHarvester (Google, Bing, Yahoo, crtsh, OTX, urlscan)
  ├── Secret Scanning   → trufflehog (git, filesystem, S3)
  ├── Git Leaks         → gitleaks detect on dumped repos
  ├── GitHub Recon      → gitrob organisation-level secret hunting
  └── S3/GCS Buckets    → 13 naming variants × AWS + GCP endpoints
```

Findings: Leaked credentials, API keys in code, misconfigured cloud buckets, employee emails

---

Phase 9 — Exploit Research & CVE Correlation 💣

Tools: searchsploit msfconsole

```
Services + CVEs
  ├── Per-Service Lookup  → searchsploit JSON for every nmap service
  ├── CVE Correlation     → nuclei CVEs → searchsploit → Metasploit modules
  ├── CMS Exploits        → detected CMS → exploit-db search
  └── MSF Module Mapping  → service names → high-rank exploit/auxiliary modules
      └── Print: rank, path, date, description
```

Output: Exploit chains: `Service Version → CVE → ExploitDB ID → Metasploit Module`

Phase 10 — Advanced Attacks 🔬

Techniques:

Attack	Description	Severity
Race Condition	25 simultaneous requests to coupon/redeem/transfer endpoints	HIGH
HTTP Parameter Pollution	Duplicate parameters with injected values	MEDIUM
Auth Bypass (Headers)	X-Original-URL, X-Forwarded-For, X-Custom-IP-Authorization on 401/403	CRITICAL
Auth Bypass (Path)	Case manipulation, URL encoding, semicolon injection, double slashes	CRITICAL
Default Credentials	15 credential pairs tested against all detected login panels	CRITICAL
JWT Attacks	alg:none, weak HS256, sensitive data in payload	CRITICAL/HIGH
HTTP Request Smuggling	CL.TE and TE.CL probes with timing analysis	HIGH

📖 Usage

Basic Scan

```bash
python3 lx-bot.py -t https://target.com
```

With Burp Suite Proxy (ALL Traffic Intercepted)

```bash
# 1. Start Burp Suite — configure proxy listener on 127.0.0.1:8080
# 2. Enable "Intercept is off" to let traffic flow through
python3 lx-bot.py -t https://target.com --proxy http://127.0.0.1:8080
```

All curl commands, httpx, dalfox, sqlmap, nuclei, and every other tool will route through your Burp proxy automatically.

Target Specific Phases Only

```bash
# Recon + ports + web analysis only
python3 lx-bot.py -t https://target.com --only-phases 1,2,3

# Injection testing only (requires prior recon data)
python3 lx-bot.py -t https://target.com --only-phases 7

# Nuclei + OSINT + Exploits
python3 lx-bot.py -t https://target.com --only-phases 6,8,9
```

Fast Scan (Skip Heavy Tools)

```bash
python3 lx-bot.py -t https://target.com --skip-heavy --threads 25
```

Full Enterprise Scan (Maximum Coverage)

```bash
sudo python3 lx-bot.py -t https://target.com \
  --full \
  --threads 30 \
  --wpscan-api YOUR_WPSCAN_API_KEY \
  --github-token YOUR_GITHUB_TOKEN \
  -o /opt/engagements/target-2026/
```

Custom Output Directory

```bash
```

```
python3 lx-bot.py -t https://target.com -o /path/to/engagement/
```

Check Tool Status

bash

```
python3 lx-bot.py --check-tools
```

Install All Tools

bash

```
python3 lx-bot.py --install-tools
# OR directly:
python3 resource_manager.py --install
```

⚙️ CLI Reference

```
usage: lx-bot [-h] -t URL [--threads N] [--only-phases 1,2,3] [--skip-heavy]
              [--full] [--proxy URL] [-o DIR] [--wpscan-api KEY]
              [--github-token TOKEN] [--check-tools] [--install-tools]
              [--version]

LX-BOT ULTIMATE v5.0 – Next-Gen Enterprise Penetration Testing

required arguments:
  -t, --target URL        Target URL or domain (e.g. https://target.com)

Scan Control:
  --threads N, -T N       Concurrent threads (default: 20)
  --only-phases 1,2,3     Run specific phases only (comma-separated)
  --skip-heavy            Skip heavy/slow tools (amass, nikto, commix)
  --full                  Run all 10 phases with maximum coverage

Proxy:
  --proxy URL, -P URL     HTTP proxy for Burp Suite (http://127.0.0.1:8080)

Output:
  -o, --output DIR        Output directory (default: ./lx-bot-results/)

API Keys & Tokens:
```

```
  --wpscan-api KEY        WPScan API token for vulnerability database
  --github-token TOKEN    GitHub personal access token for OSINT

Utility:
  --check-tools           Show tool installation status and exit
  --install-tools         Install all missing tools and exit
  --version               Show version number and exit
```

📁 Output Structure

```
lx-bot-results/
└── target.com/
    └── 20260213_143022/
        │
        ├── target.com_report.html       ← Professional HTML report (graphs + CVSS)
        ├── target.com_results.json      ← Complete raw JSON results
        │
        ├── subdomains.txt               ← All discovered subdomains (deduped)
        ├── live.txt                     ← Live hosts with status/title/tech
        ├── live_urls.txt                ← Plain URLs list for tool input
        ├── targets.txt                  ← Probe targets list
        │
        ├── nmap.xml                     ← Nmap XML results (parseable)
        ├── nmap.gnmap                   ← Nmap grepable output
        ├── masscan.json                 ← Masscan JSON port data
        │
        ├── nuclei_critical_high.jsonl   ← Critical & high Nuclei findings
        ├── nuclei_medium.jsonl          ← Medium/low Nuclei findings
        ├── nuclei_cves.jsonl            ← CVE-specific Nuclei results
        ├── nuclei_exposures.jsonl       ← Exposure template results
        ├── nuclei_misconfig.jsonl       ← Misconfiguration findings
        │
        ├── wpscan.json                  ← WordPress scan (if CMS detected)
        ├── ffuf.json                    ← Directory fuzzing results
        ├── feroxbuster.txt              ← Recursive content discovery
        ├── dirsearch.json               ← Directory search output
        ├── katana.txt                   ← Crawled URL list
        │
        ├── amass.txt                    ← OWASP Amass subdomain output
        ├── findomain.txt                ← Findomain output
        ├── harvester.json               ← theHarvester OSINT data
```

```
├── gitleaks.json              ← Gitleaks secret findings
│
├── ssl.json                   ← testssl.sh findings
├── sslscan.xml                ← SSLScan XML output
├── takeover_subjack.txt       ← Subdomain takeover candidates
│
├── sqlmap/                    ← SQLmap session & results
├── git_dump/                  ← Dumped git repository (if found)
│
└── screenshots/               ← Visual reconnaissance
    ├── target_com.png
    ├── api_target_com.png
    └── ...
```

📊 Report Features

The HTML report (`report_generator.py`) includes:

Cover Page

- Target details, scan timestamp, engagement metadata
- Risk level indicator with colour coding

Executive Summary

- Plain-English risk assessment for C-level presentation
- Attack surface overview
- Top 5 critical findings with business impact

Interactive Graphs

- **Severity Distribution** — Donut chart (Critical/High/Medium/Low/Info)
- **CVSS Score Histogram** — Distribution of all finding scores
- **Phase Coverage Map** — Radar chart of attack surface tested
- **Timeline** — Finding discovery over scan duration

Vulnerability Details (per finding)

- CVSS v3.1 score + vector breakdown
- OWASP Top 10 2021 mapping

- CWE classification
- Proof-of-concept / reproduction steps
- Metasploit module (if available)
- Remediation recommendation

Remediation Roadmap

- Priority matrix (Impact × Likelihood)
- Timeline recommendations (Immediate / 30 days / 90 days)
- Fix verification checklist

Compliance Mapping

- OWASP Top 10 2021 coverage
- PCI-DSS requirement mapping
- ISO 27001 control mapping

🔧 Configuration

Environment Variables

```bash
# WPScan vulnerability database (free at https://wpscan.com/register)
export WPSCAN_API="your-api-key-here"

# GitHub token for OSINT (Settings → Developer Settings → Personal Access Tokens)
export GITHUB_TOKEN="ghp_xxxxxxxxxxxxxxxxxxxx"

# Shodan API key for enhanced service intelligence
export SHODAN_API_KEY="your-shodan-key"

# Burp Suite proxy (alternative to --proxy flag)
export HTTP_PROXY="http://127.0.0.1:8080"
export HTTPS_PROXY="http://127.0.0.1:8080"
```

Wordlist Configuration

LX-BOT automatically detects wordlists in this priority order:

```
1. /usr/share/seclists/Discovery/Web-Content/common.txt
2. /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
3. /usr/share/wordlists/dirb/common.txt
4. /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
5. /opt/SecLists/Discovery/Web-Content/common.txt
6. ~/SecLists/Discovery/Web-Content/common.txt
```

Install SecLists for best results:

```bash
sudo git clone --depth 1 https://github.com/danielmiessler/SecLists /usr/share/seclists
```

Thread Tuning

Scan Type	Recommended Threads	Use Case
`--threads 10`	Conservative	Fragile targets, strict rate limiting
`--threads 20`	Default	Most enterprise targets
`--threads 30`	Aggressive	Resilient targets, lab environments
`--threads 50`	Maximum	Internal networks, explicit permission

📦 Installation

Prerequisites

```bash
```

```
# Kali Linux / Parrot OS (recommended)
sudo apt update && sudo apt install -y \
  python3 python3-pip git curl wget nmap nikto \
  golang-go ruby ruby-dev perl libssl-dev

# Install Go (for go-based tools)
python3 resource_manager.py --install

# Verify Go path
echo 'export PATH=$PATH:$(go env GOPATH)/bin' >> ~/.bashrc
source ~/.bashrc
```

Python Dependencies

```bash
# Standard install
pip install -r requirements.txt

# Kali / Parrot / Ubuntu (override system packages)
pip install -r requirements.txt --break-system-packages

# Virtual environment (isolated install)
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Tool Installation

```bash
# Install all 60+ tools automatically
python3 resource_manager.py --install

# Check status afterwards
python3 resource_manager.py --check

# Check via lx-bot
python3 lx-bot.py --check-tools
```

Docker (optional)

```dockerfile
```

```
FROM kalilinux/kali-rolling
RUN apt update && apt install -y python3 python3-pip git curl wget nmap golang-go ruby ruby-dev
COPY . /opt/lx-bot/
WORKDIR /opt/lx-bot
RUN pip install -r requirements.txt --break-system-packages
RUN python3 resource_manager.py --install
ENTRYPOINT ["python3", "lx-bot.py"]
```

bash

```
docker build -t lx-bot .
docker run --rm lx-bot -t https://target.com
```

🐛 Troubleshooting

Issue: Tools Not Found

bash

```
# Check what's missing
python3 lx-bot.py --check-tools

# Install missing tools
python3 resource_manager.py --install

# Verify Go PATH
echo $PATH | grep go
export PATH=$PATH:~/go/bin:~/.local/bin
```

Issue: Masscan Requires Root

bash

```
# Masscan needs raw socket access
sudo python3 lx-bot.py -t https://target.com

# OR use rustscan instead (no root required)
# rustscan is auto-preferred over masscan when available
```

Issue: Proxy Not Working

```bash
# 1. Ensure Burp Suite is running and proxy listener is active
# 2. Verify proxy listener is on the correct port
# 3. Use this exact format:
python3 lx-bot.py -t https://target.com --proxy http://127.0.0.1:8080

# 4. For HTTPS targets, ensure Burp CA cert is installed or use -k flag
# 5. Test proxy manually:
curl -x http://127.0.0.1:8080 -k https://target.com
```

Issue: Nuclei Templates Missing

```bash
# Update nuclei templates
nuclei -update-templates

# Manual template download
git clone https://github.com/projectdiscovery/nuclei-templates ~/.local/nuclei-templates
```

Issue: WPScan API Rate Limiting

```bash
# Register for free API at https://wpscan.com/register
# Set your token:
export WPSCAN_API="your-token-here"
# OR use --wpscan-api flag
python3 lx-bot.py -t https://wordpress-target.com --wpscan-api YOUR_TOKEN
```

Issue: Slow Scan Performance

```bash
```

```
# Increase threads (check target capacity first)
python3 lx-bot.py -t https://target.com --threads 30

# Skip slow tools for faster coverage
python3 lx-bot.py -t https://target.com --skip-heavy

# Run only essential phases
python3 lx-bot.py -t https://target.com --only-phases 1,2,6,7
```

Issue: Permission Errors on Output

```
bash

# Specify a writable directory
python3 lx-bot.py -t https://target.com -o /tmp/scan-results/
```

🔬 Advanced Use Cases

Bug Bounty Workflow

```
bash

# Phase 1 only — fast subdomain collection
python3 lx-bot.py -t https://bugbounty-target.com --only-phases 1

# Phase 4+5 — endpoint and content discovery
python3 lx-bot.py -t https://bugbounty-target.com --only-phases 4,5

# Phase 7 — injection testing on discovered endpoints
python3 lx-bot.py -t https://bugbounty-target.com --only-phases 7
```

Red Team Engagement

```
bash
```

```
# Full engagement with Burp Suite integration
python3 lx-bot.py -t https://client.com \
  --proxy http://127.0.0.1:8080 \
  --full --threads 20 \
  --wpscan-api $WPSCAN_API \
  --github-token $GITHUB_TOKEN \
  -o /opt/engagement/client-2026/
```

Internal Network Assessment

bash

```
# Fast internal scan (no rate limits)
sudo python3 lx-bot.py -t http://192.168.1.0/24 \
  --threads 50 \
  --only-phases 1,2,3,6 \
  -o /opt/internal-scan/
```

CI/CD Integration

bash

```
#!/bin/bash
# security-scan.sh — run in pipeline on staging deployments

python3 lx-bot.py \
  -t "$STAGING_URL" \
  --only-phases 3,6,7 \
  --threads 15 \
  -o "/artifacts/security-scan-$(date +%Y%m%d)" 2>&1

# Check for critical findings
CRITICAL=$(jq '.stats.critical' /artifacts/security-scan-*/*/results.json)
if [ "$CRITICAL" -gt "0" ]; then
  echo "::error::$CRITICAL critical vulnerabilities found!"
  exit 1
fi
```

📐 CVSS v3.1 Scoring

LX-BOT automatically calculates CVSS v3.1 Base Scores for all findings:

Score	Severity	Colour	Action
9.0 – 10.0	**CRITICAL**	🔴 Red	Immediate remediation required
7.0 – 8.9	**HIGH**	🟠 Orange	Fix within 7 days
4.0 – 6.9	**MEDIUM**	🟡 Yellow	Fix within 30 days
0.1 – 3.9	**LOW**	🟢 Green	Fix within 90 days
0.0	**NONE**	⚪ Grey	Informational

Vulnerability Type → CVSS Inference

Vulnerability	Default CVSS Vector	Typical Score
RCE / Command Injection	AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H	**9.8 Critical**
SQL Injection	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L	**9.1 Critical**
SSTI	AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H	**9.8 Critical**
SSRF	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L	**8.2 High**
XSS (Stored)	AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N	**6.1 Medium**
XXE	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L	**8.2 High**
Open Redirect	AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N	**6.1 Medium**
Missing HSTS	AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	**3.1 Low**

🔒 OWASP Top 10 2021 Coverage

OWASP Category	LX-BOT Coverage	Phase
A01 – Broken Access Control	IDOR, Path Traversal, Auth Bypass, CORS	7, 10
A02 – Cryptographic Failures	SSL/TLS, Secrets in JS, HTTP (no HTTPS)	3, 4, 8
A03 – Injection	SQLi, XSS, CMDi, SSTI, XXE	7

OWASP Category	LX-BOT Coverage	Phase
A04 – Insecure Design	Race Conditions, Business Logic	10
A05 – Security Misconfiguration	Headers, CORS, Git Exposure, Debug Endpoints	3, 5
A06 – Vulnerable Components	CVE Correlation, Nuclei CVE Templates	6, 9
A07 – Authentication Failures	Default Creds, JWT Attacks, Auth Bypass	10
A08 – Integrity Failures	JWT alg:none, Deserialization probes	10
A09 – Logging Failures	Exposed logs, debug endpoints	5
A10 – Server-Side Request Forgery	SSRF with cloud metadata	7

🤝 Contributing

1. Fork the repository
2. Create a feature branch: `git checkout -b feature/new-phase`
3. Write tests for new functionality
4. Submit a pull request with description

Adding a New Tool

1. Add tool definition in `resource_manager.py` `self.tools` dict
2. Add tool execution method in the appropriate phase class in `lx.py` or `lx-bot.py`
3. Add findings to `ResultsStore` via `store.add_vuln()` or `store.add()`
4. Update `README.md` tool table

Adding a New Vulnerability Check

```python
```

```
async def _my_new_check(self, url: str):
    rc, out, _ = await self.runner.run(
        f'my-tool --target {url} --output json',
        timeout=120, tag='my-tool'
    )
    if 'VULNERABLE' in out:
        self.store.add_vuln({
            'name':        'My New Vulnerability',
            'severity':    'high',          # critical/high/medium/low/info
            'url':         url,
            'description': 'Description of the issue',
            'tool':        'my-tool',
            'owasp':       'A03:2021',      # optional
            'cve':         ['CVE-2024-XXXX'],  # optional
        })
        print_finding('high', 'My New Vulnerability!', url)
```

📜 Changelog

v5.0 (2026-02-16) — Current

- Complete rewrite with async engine
- Split into `lx.py` (core) + `lx-bot.py` (orchestrator) — 4,600+ lines total
- Added Phase 10 advanced attacks (race conditions, JWT, smuggling)
- Metasploit integration with module auto-mapping
- CVSS v3.1 calculator with vector inference
- Interactive Plotly report with executive summary
- WAF-aware fuzzing rate limiting
- 16-pattern JS secret scanner
- S3/GCS bucket enumeration (13 naming variants)

v4.0

- Added GraphQL introspection testing
- Nuclei template auto-update before scan
- CORS + cookie security header analysis
- git-dumper integration on .git exposure

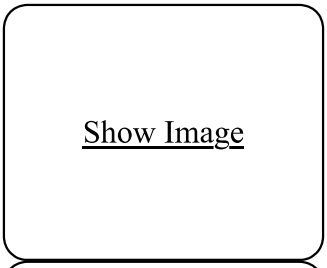
v3.0

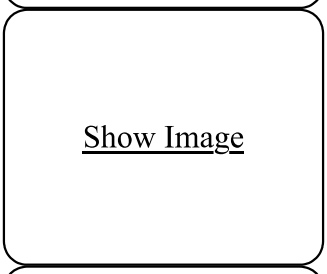
- Multi-tool CMS scanning (wpscan, joomscan, droopescan, magescan)
- testssl.sh TLS audit integration
- Wayback CDX API for historical URL collection

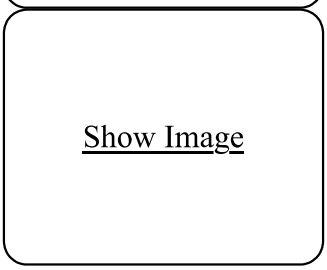
License

LX-BOT ULTIMATE v5.0 — Built for Professional Security Testing

Show Image







For authorized security testing only. Always get written permission.