John the Ripper: Master Password Cracking Tool

From Hash Cracking to SSH Key Recovery

Introduction

John the Ripper, commonly shortened to 'John,' stands as one of the most powerful and versatile password cracking tools in cybersecurity. Since its creation by Solar Designer in 1996, John has evolved into an essential tool for security professionals, penetration testers, and CTF competitors worldwide. Its ability to crack everything from simple MD5 hashes to complex SSH private keys makes it indispensable for both offensive security assessments and defensive password auditing.

Unlike GPU-accelerated tools like Hashcat, John the Ripper leverages CPU power and implements intelligent cracking strategies including dictionary attacks, rule-based mangling, and single crack mode. This comprehensive guide explores John's capabilities through practical examples, demonstrating how to crack Windows authentication hashes, Linux shadow files, password-protected archives, and SSH keys.

Learning Objectives

This comprehensive guide covers:

- Cracking Windows NTLM authentication hashes
- Cracking Linux /etc/shadow password hashes
- Cracking password-protected ZIP and RAR archives
- Cracking SSH private key passphrases
- Understanding single crack mode and word mangling techniques

Understanding Hash Functions

Before diving into practical cracking, understanding hash fundamentals proves essential. A hash represents data of any length in fixed-length form, masking the original value through mathematical transformation.

Hash Examples

Example 1: Short input

- Input: `polo` (4 characters)
- MD5 Hash: `b53759f3ce692de7aff1b5779d3964da` (32 characters)

Example 2: Longer input

- Input: `polomints` (9 characters)
- MD5 Hash: `584b6e4f4586e136bc280f27f9c64f3b` (32 characters)

Key Observation: Both inputs-despite different lengths (4 vs 9 characters)-produce 32-character MD5 hashes. This fixed output size regardless of input length defines hash functions.

Why Hashes Are Secure

Hash functions operate as one-way functions-easy to compute forward, computationally infeasible to reverse. This asymmetry stems from fundamental computer science complexity theory:

- **P (Polynomial Time):** Problems solvable in reasonable time. Hashing belongs here-calculating hashes executes quickly
- **NP (Non-deterministic Polynomial Time):** Problems where solutions can be verified quickly but finding them proves computationally intractable. Reversing hashes belongs here

The P vs NP Problem: While hashing algorithms execute in polynomial time (P), no known algorithm can efficiently reverse them. This mathematical property-rooted in the famous P vs NP problem-makes hash functions cryptographically secure.

Where John Comes In

Although mathematically irreversible, hashes remain vulnerable to dictionary attacks. If you possess a hash and know the algorithm, you can hash thousands or millions of potential passwords and compare results. When matches occur, you've cracked the password.

John the Ripper excels at this process, implementing optimized algorithms for rapid dictionary and brute-force attacks across numerous hash types. Its CPU-based approach works effectively in virtual machines and supports intelligent cracking strategies beyond simple dictionary attacks.

John the Ripper: Basic Usage

Understanding John's syntax and operation modes provides the foundation for effective hash cracking. John offers multiple approaches depending on hash type and available information.

Basic Command Syntax

```
john [options] [filepath]
```

- `john` - Invokes the John the Ripper program
- `[options]` - Specifies cracking mode and parameters
- `[filepath]` - File containing hashes to crack

Automatic Hash Detection and Cracking

John includes built-in hash type detection, attempting to identify and crack hashes automatically. While convenient, this approach proves unreliable for complex or uncommon hash formats.

Basic automatic cracking:

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash_file.txt
```

Command breakdown:

- `--wordlist=` - Specifies dictionary attack mode
- `/usr/share/wordlists/rockyou.txt` - Path to password wordlist (14M+ passwords)

Hash Identification

When John's automatic detection fails, manual hash identification becomes necessary. Several tools facilitate this process.

Using hash-identifier

The hash-identifier Python tool provides quick hash type identification:

Step 1: Download hash-identifier

```
wget https://gitlab.com/kalilinux/packages/hash-identifier/-/raw/kali/master/hash-id.py
```

Step 2: Run hash-identifier

```
python3 hash-id.py
```

Step 3: Enter hash when prompted

```
HASH: 2e728dd31fb5949bc39cac5a9f066498
```

Output:

- Possible Hashs: [+] MD5
- [+] Domain Cached Credentials

Online Hash Identifiers

→ **Hashes.com Identifier:** https://hashes.com/en/tools/hash_identifier
→ **TunnelsUP Hash Analyzer:** https://www.tunnelsup.com/hash-analyzer/

Format-Specific Cracking

Once hash type is identified, specify the format explicitly for reliable cracking:

```
john --format=[format] --wordlist=[wordlist_path] [hash_file]
```

Example: Cracking MD5 hash

```
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
```

Important Note: Standard hash types (MD5, SHA1, SHA256) require the 'raw-' prefix. Use `john --list=formats` to view all supported formats or `john --list=formats | grep -iF 'md5'` to search specific formats.

Cracking Windows NTLM Hashes

NTLM (NT LAN Manager) represents the hash format modern Windows systems use for storing user and service passwords. Understanding NTLM cracking proves essential for Windows penetration testing and red team operations.

Understanding NTLM

NT designation originally meant 'New Technology' when Microsoft introduced Windows NT-the first Windows version not built on MS-DOS. While Microsoft dropped the NT naming from consumer products, it persists in technical nomenclature like NTLM.

Windows stores these hashes in SAM (Security Account Manager) database. Obtaining NTLM hashes requires privileged access, typically achieved through:

- Dumping the SAM database
- Using Mimikatz for memory extraction
- Accessing Active Directory database (NTDS.dit)

Practical NTLM Cracking Example

Example NTLM hash:

```
5460C85BD858A11475115D2DD3A82333
```

Cracking command:

```
john --format=nt --wordlist=/usr/share/wordlists/rockyou.txt ntlm_hash.txt
```

Alternative: Pass-the-Hash Attack - Sometimes you don't need to crack NTLM hashes. Pass-the-hash attacks use the hash directly for authentication, bypassing password cracking entirely. However, weak password policies often make cracking viable and valuable for lateral movement.

Cracking Linux /etc/shadow Hashes

Linux stores password hashes in /etc/shadow, readable only by root. Successfully cracking these hashes provides powerful privilege escalation opportunities during penetration tests.

Understanding /etc/shadow Format

Each line in /etc/shadow contains one user account with colon-separated fields:

```
root:$6$salt$hash:18576:0:99999:7:::
```

1. Username (root)
2. Encrypted password (6salt$hash)
3. Last password change date
4. Additional password aging information

The Unshadowing Process

John requires combining /etc/shadow with /etc/passwd using the unshadow tool. This merges username and hash information into a format John understands.

Step 1: Prepare Files

File: local_passwd

```
root:x:0:0::/root:/bin/bash
```

File: local_shadow

```
root:$6$2nwjN454g.dv4HN/$m9Z/r2xVfweYVkrr.v5Ft8Ws3/YYksfNwq96UL1FX0OJjY1L6l.DS3KEVsZ9rOVLB/ldTeEL/OIh
JZ4GMFMGA0:18576:::::
```

Step 2: Run Unshadow

```
unshadow local_passwd local_shadow > unshadowed.txt
```

Command breakdown:

- `unshadow` - Merging tool
- `local_passwd` - /etc/passwd copy
- `local_shadow` - /etc/shadow copy
- `> unshadowed.txt` - Redirect output to file

Step 3: Crack with John

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=sha512crypt unshadowed.txt
```

Note: The 6 prefix indicates sha512crypt. While John sometimes auto-detects this, explicitly specifying --format=sha512crypt ensures reliable cracking.

Single Crack Mode: Word Mangling

Single Crack mode represents John's intelligent cracking strategy, using username information to generate targeted wordlists through word mangling-systematically modifying usernames into probable passwords.

Understanding Word Mangling

Consider username 'Markus'. John generates variations exploiting common password patterns:

- **Number appending:** Markus1, Markus2, Markus3, Markus123
- **Capitalization variants:** MArkus, MARkus, MARKus, MARKUS
- **Special character appending:** Markus!, Markus$, Markus*, Markus@
- **Combined variations:** Markus1!, MARKUS123, markus@2024

This technique exploits human tendencies toward predictable passwords based on personal information.

GECOS Field Integration

GECOS (General Electric Comprehensive Operating System) refers to the fifth field in /etc/passwd, storing user information like full name, office number, and phone number. John extracts this data for additional mangling material.

Example /etc/passwd entry:

```
mike:x:1001:1001:Mike Johnson,Room 405,555-1234:/home/mike:/bin/bash
```

John extracts 'Mike Johnson', 'mike', and generates variants like: Mike1, Johnson123, MikeJ!, etc.

Using Single Crack Mode

```
john --single --format=[format] [hash_file]
```

Example:

```
john --single --format=raw-sha256 hashes.txt
```

Critical File Format Requirement: Single crack mode requires username-prefixed hashes:

Incorrect format:

```
1efee03cdcb96d90ad48ccc7b8666033
```

Correct format:

```
mike:1efee03cdcb96d90ad48ccc7b8666033
```

Cracking Password-Protected Archives

John extends beyond simple hash cracking to password-protected file archives. CTF challenges and real-world scenarios frequently involve ZIP and RAR files requiring password recovery.

Cracking ZIP Files

The zip2john utility extracts hash information from password-protected ZIP archives, converting them to John-compatible format.

Step 1: Extract Hash from ZIP

```
zip2john secure_file.zip > zip_hash.txt
```

Command breakdown:

- `zip2john` - Conversion tool
- `secure_file.zip` - Password-protected ZIP
- `> zip_hash.txt` - Output file containing hash

Step 2: Crack with John

```
john --wordlist=/usr/share/wordlists/rockyou.txt zip_hash.txt
```

Practical Example:

Imagine you've found 'confidential.zip' during penetration testing. Extract hash with zip2john, crack with rockyou.txt wordlist, then unzip using recovered password to access contents.

Cracking RAR Files

RAR archives, created by WinRAR, follow identical process using rar2john converter.

Step 1: Extract Hash from RAR

```
/opt/john/rar2john secure_archive.rar > rar_hash.txt
```

Note: rar2john location varies by distribution. On Kali Linux, it's typically in /opt/john/ or /usr/sbin/

Step 2: Crack with John

```
john --wordlist=/usr/share/wordlists/rockyou.txt rar_hash.txt
```

Cracking SSH Private Keys

SSH supports key-based authentication using private keys (id_rsa) instead of passwords. When private keys are passphrase-protected, John can crack these passphrases, enabling SSH access during penetration tests.

Understanding SSH Key Authentication

Standard SSH authentication uses passwords. Key-based authentication offers stronger security through cryptographic key pairs-a private key (kept secret) and public key (shared with servers). The private key often requires a passphrase for additional protection.

Common Scenario: You've found an id_rsa file on a compromised system or in a backup. It's passphrase-protected. Cracking the passphrase grants SSH access to the associated account.

Using ssh2john

Step 1: Convert SSH Key to John Format

```
python3 /opt/john/ssh2john.py id_rsa > ssh_hash.txt
```

Alternative paths:

- Kali Linux: `python /usr/share/john/ssh2john.py`
- AttackBox: `python3 /opt/john/ssh2john.py`

Step 2: Crack with John

```
john --wordlist=/usr/share/wordlists/rockyou.txt ssh_hash.txt
```

Step 3: Use Cracked Passphrase

Once John cracks the passphrase, use it with the private key for SSH authentication:

```
ssh -i id_rsa user@target_ip
```

When prompted, enter the cracked passphrase to authenticate.

Quick Reference Command Table

Task	Command
Basic cracking	`john --wordlist=rockyou.txt hash.txt`
Format-specific	`john --format=raw-md5 --wordlist=rockyou.txt hash.txt`
List formats	`john --list=formats`
Unshadow Linux	`unshadow passwd shadow > unshadowed.txt`
Single crack mode	`john --single --format=raw-sha256 hash.txt`
ZIP cracking	`zip2john file.zip > hash.txt && john hash.txt`
RAR cracking	`rar2john file.rar > hash.txt && john hash.txt`
SSH key cracking	`ssh2john.py id_rsa > hash.txt && john hash.txt`
Show cracked	`john --show hash.txt`

External Resources

→ **John the Ripper Official:** https://www.openwall.com/john/
→ **John Documentation:** https://github.com/openwall/john/blob/bleeding-jumbo/doc/
→ **Hash-Identifier Tool:** https://gitlab.com/kalilinux/packages/hash-identifier
→ **Wordlists (SecLists):** https://github.com/danielmiessler/SecLists

Key Takeaways

- John the Ripper excels at CPU-based password cracking across numerous hash types
- Automatic detection works for common hashes; format specification ensures reliability
- Windows NTLM hashes require --format=nt specification
- Linux /etc/shadow requires unshadow tool combining passwd and shadow files
- Single crack mode uses username information for intelligent word mangling
- Archive cracking (ZIP/RAR) uses conversion tools: zip2john, rar2john
- SSH private key cracking uses ssh2john.py for passphrase recovery

Conclusion

John the Ripper remains the definitive CPU-based password cracking tool, combining versatility with sophisticated cracking strategies. From basic dictionary attacks against MD5 hashes to intelligent single crack mode exploiting username patterns, John adapts to diverse security assessment scenarios.

Understanding John's capabilities-NTLM cracking for Windows penetration testing, /etc/shadow cracking for Linux privilege escalation, archive password recovery for data access, and SSH key passphrase cracking for remote authentication-provides essential skills for cybersecurity professionals. Whether conducting authorized penetration tests, competing in CTF challenges, or auditing organizational password policies, John the Ripper delivers the power and flexibility required for effective password security assessment.

Continue mastering password cracking through hands-on practice with real-world scenarios and CTF challenges.