Liber8tion Cracker

```
#!/usr/bin/env python3

import os
import argparse
import subprocess
import sys
import tempfile
import shutil

def run_hashcat(cmd, description):
    """Run a hashcat command with proper logging"""
    print(f"[+] {description}")
    print(f"[+] Command: {' '.join(cmd)}")
    try:
        subprocess.run(cmd, check=False)
    except Exception as e:
        print(f"[-] Error running hashcat: {e}")

def main():
    parser = argparse.ArgumentParser(description='Crack hashes using the Liber8ion Passphrase
Standard')
    parser.add_argument('--hash-file', required=True, help='File containing hashes to crack')
    parser.add_argument('--hash-type', required=True, help='Hashcat hash type (e.g. 0 for MD5,
100 for SHA1)')
    parser.add_argument('--wordlist', default='/usr/share/wordlists/rockyou.txt',
help='Dictionary wordlist')
    parser.add_argument('--output', default='cracked_passwords.txt', help='Output file for
cracked passwords')
    args = parser.parse_args()

    # Create temporary directory
    temp_dir = tempfile.mkdtemp(prefix="liber8ion_")
    print(f"[+] Using temporary directory: {temp_dir}")

    # Path for the potfile
    potfile = os.path.join(temp_dir, "liber8ion.potfile")
```

```
    # Create a smaller dictionary with lowercase words
    print(f"[+] Creating optimized wordlist from {args.wordlist}...")
    lowercase_dict = os.path.join(temp_dir, "lowercase_dict.txt")
    uppercase_dict = os.path.join(temp_dir, "uppercase_dict.txt")

    try:
        # Take a reasonable subset to avoid memory issues
        with open(args.wordlist, 'r', encoding='latin-1', errors='ignore') as infile, \
             open(lowercase_dict, 'w') as lower_out, \
             open(uppercase_dict, 'w') as upper_out:
            for i, line in enumerate(infile):
                if i >= 100000:  # Limit to first 100k words
                    break
                word = line.strip()
                if word and len(word) >= 3 and len(word) <= 10:  # Filter reasonable word
lengths
                    lower_out.write(f"{word.lower()}\n")
                    upper_out.write(f"{word.upper()}\n")
    except Exception as e:
        print(f"[-] Error processing wordlist: {e}")
        sys.exit(1)

    # Create file with digits
    digits_dict = os.path.join(temp_dir, "digits.txt")
    with open(digits_dict, 'w') as f:
        for i in range(10):
            f.write(f"{i}\n")

    # Create special character dictionaries
    hyphen_dict = os.path.join(temp_dir, "hyphen.txt")
    with open(hyphen_dict, 'w') as f:
        f.write("-\n")

    special_chars_dict = os.path.join(temp_dir, "special_chars.txt")
    with open(special_chars_dict, 'w') as f:
        for c in "!@#$%^&*()-_=+[]{}|;:,.<>?/":
            f.write(f"{c}\n")

    # Create liber8 file
    liber8_dict = os.path.join(temp_dir, "liber8.txt")
```

```
    with open(liber8_dict, 'w') as f:
        f.write("liber8\n")

    # Generate specific pattern dictionaries for each type
    print("[+] Generating pattern dictionaries...")

    # For Type 1 (All lowercase, hyphen separator)
    type1_patterns = os.path.join(temp_dir, "type1_patterns.txt")
    try:
        with open(lowercase_dict, 'r') as word_file, open(type1_patterns, 'w') as out_file:
            words = [w.strip() for w in word_file.readlines()]
            for word in words[:1000]:  # Limit to first 1000 words for efficient processing
                out_file.write(f"{word}-liber8-\n")
    except Exception as e:
        print(f"[-] Error generating Type 1 patterns: {e}")

    # Generate all types of patterns with special characters
    # For Types 2, 3, and 4
    special_chars = "!@#$%^&*()-_=+[]{}|;:,.<>?/"

    # Type 2 (All lowercase, any special char)
    type2_patterns = os.path.join(temp_dir, "type2_patterns.txt")
    try:
        with open(lowercase_dict, 'r') as word_file, open(type2_patterns, 'w') as out_file:
            words = [w.strip() for w in word_file.readlines()]
            for word in words[:500]:  # Limit to 500 words
                for special_char in special_chars:
                    out_file.write(f"{word}{special_char}liber8{special_char}\n")
    except Exception as e:
        print(f"[-] Error generating Type 2 patterns: {e}")

    # Type 3 lower patterns (lowercase first word, any special char)
    type3_lower_patterns = os.path.join(temp_dir, "type3_lower_patterns.txt")
    try:
        with open(lowercase_dict, 'r') as word_file, open(type3_lower_patterns, 'w') as
out_file:
            words = [w.strip() for w in word_file.readlines()]
            for word in words[:500]:  # Limit to 500 words
                for special_char in special_chars:
                    out_file.write(f"{word}{special_char}liber8{special_char}\n")
    except Exception as e:
```

```
        print(f"[-] Error generating Type 3 lower patterns: {e}")

    # Type 3 upper patterns (uppercase first word, any special char)
    type3_upper_patterns = os.path.join(temp_dir, "type3_upper_patterns.txt")
    try:
        with open(uppercase_dict, 'r') as word_file, open(type3_upper_patterns, 'w') as
out_file:
            words = [w.strip() for w in word_file.readlines()]
            for word in words[:500]:  # Limit to 500 words
                for special_char in special_chars:
                    out_file.write(f"{word}{special_char}liber8{special_char}\n")
    except Exception as e:
        print(f"[-] Error generating Type 3 upper patterns: {e}")

    # Type 4 digit patterns - with digits at end of first word
    type4_first_digit_patterns = os.path.join(temp_dir, "type4_first_digit_patterns.txt")
    try:
        with open(lowercase_dict, 'r') as word_file, open(type4_first_digit_patterns, 'w') as
out_file:
            words = [w.strip() for w in word_file.readlines()]
            for word in words[:300]:  # Limit words
                for digit in range(10):
                    for special_char in special_chars[:5]:  # Limit special chars
                        out_file.write(f"{word}{digit}{special_char}liber8{special_char}\n")
    except Exception as e:
        print(f"[-] Error generating Type 4 first word digit patterns: {e}")

    print("\n[+] Starting hash cracking with Liber8ion Passphrase Standard...")

    # Type 1: word1-liber8-word2 (all lowercase, hyphen separators)
    print("\n[+] Cracking Type 1 passphrases...")
    cmd = [
        "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
        type1_patterns, lowercase_dict,
        "--potfile-path", potfile
    ]
    run_hashcat(cmd, "Trying Type 1 patterns: word1-liber8-word2 (all lowercase)")

    # Type 2: word1<special>liber8<special>word2 (all lowercase)
    print("\n[+] Cracking Type 2 passphrases...")
    cmd = [
```

```
        "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
        type2_patterns, lowercase_dict,
        "--potfile-path", potfile
    ]
    run_hashcat(cmd, "Trying Type 2 patterns: word1<special>liber8<special>word2 (all
lowercase)")

    # Type 3: Each word all lowercase OR all uppercase
    print("\n[+] Cracking Type 3 passphrases - lowercase first word...")
    cmd = [
        "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
        type3_lower_patterns, lowercase_dict,
        "--potfile-path", potfile
    ]
    run_hashcat(cmd, "Trying Type 3 patterns: lower<special>liber8<special>lower")

    cmd = [
        "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
        type3_lower_patterns, uppercase_dict,
        "--potfile-path", potfile
    ]
    run_hashcat(cmd, "Trying Type 3 patterns: lower<special>liber8<special>UPPER")

    print("\n[+] Cracking Type 3 passphrases - uppercase first word...")
    cmd = [
        "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
        type3_upper_patterns, lowercase_dict,
        "--potfile-path", potfile
    ]
    run_hashcat(cmd, "Trying Type 3 patterns: UPPER<special>liber8<special>lower")

    cmd = [
        "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
        type3_upper_patterns, uppercase_dict,
        "--potfile-path", potfile
    ]
    run_hashcat(cmd, "Trying Type 3 patterns: UPPER<special>liber8<special>UPPER")

    # Type 4: One word with digit appended
    print("\n[+] Cracking Type 4 passphrases - first word with digit...")
    cmd = [
```

```
    "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
    type4_first_digit_patterns, lowercase_dict,
    "--potfile-path", potfile
]
run_hashcat(cmd, "Trying Type 4 patterns: word1+digit<special>liber8<special>word2")

cmd = [
    "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
    type4_first_digit_patterns, uppercase_dict,
    "--potfile-path", potfile
]
run_hashcat(cmd, "Trying Type 4 patterns: word1+digit<special>liber8<special>WORD2")

# Type 4 with second word with digit
# For this, we'll use the Type 3 patterns but with a rule to append a digit
print("\n[+] Cracking Type 4 passphrases - second word with digit...")

# Create a digit append rule file
append_digit_rule = os.path.join(temp_dir, "append_digit.rule")
with open(append_digit_rule, 'w') as f:
    for i in range(10):
        f.write(f"${i}\n")

# For lowercase second word with digit
cmd = [
    "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
    type3_lower_patterns, lowercase_dict,
    "-r", append_digit_rule,
    "--potfile-path", potfile
]
run_hashcat(cmd, "Trying Type 4 patterns: word1<special>liber8<special>word2+digit")

# For uppercase second word with digit
cmd = [
    "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
    type3_lower_patterns, uppercase_dict,
    "-r", append_digit_rule,
    "--potfile-path", potfile
]
run_hashcat(cmd, "Trying Type 4 patterns: word1<special>liber8<special>WORD2+digit")
```

```
    # Same for uppercase first words
    cmd = [
        "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
        type3_upper_patterns, lowercase_dict,
        "-r", append_digit_rule,
        "--potfile-path", potfile
    ]
    run_hashcat(cmd, "Trying Type 4 patterns: WORD1<special>liber8<special>word2+digit")

    cmd = [
        "hashcat", "-a0", f"-m{args.hash_type}", args.hash_file,
        type3_upper_patterns, uppercase_dict,
        "-r", append_digit_rule,
        "--potfile-path", potfile
    ]
    run_hashcat(cmd, "Trying Type 4 patterns: WORD1<special>liber8<special>WORD2+digit")

    # Process results to the output file
    print(f"\n[+] Processing results to {args.output}...")
    with open(potfile, 'r') as pot, open(args.output, 'w') as out:
        for line in pot:
            if ':' in line:
                hash_val, plaintext = line.strip().split(':', 1)
                out.write(f"{hash_val}:{plaintext}\n")

    print(f"\n[+] Cracking complete! Results saved to {args.output}")
    print(f"[+] To show your cracked passwords: cat {args.output}")

    # Ask if user wants to remove temp files
    response = input(f"\n[?] Remove temporary files in {temp_dir}? (y/n): ")
    if response.lower() == 'y':
        try:
            shutil.rmtree(temp_dir)
            print(f"[+] Temporary directory {temp_dir} removed")
        except Exception as e:
            print(f"[-] Error removing temporary directory: {e}")
    else:
        print(f"[+] Temporary files kept in {temp_dir}")

if __name__ == "__main__":
    main()
```

Revision #1

Created 2025-11-25 18:05:52 UTC by David Rizzo

Updated 2025-11-25 18:06:03 UTC by David Rizzo