

To brute-force the remaining part of the password based on the information provided, we can write a Python script to attempt all possible combinations of characters for the missing part.

```

Python Script:
import hashlib
import itertools

The known part of the password
known_part = "Om@nYg"

The character set based on the given password policy
characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789*%_^+_-"

The target hash
target_hash = "827d1057ad7258b180efca5e9cc25795a1a5f622"

Brute-force to find the remaining part of the password
def find_password(known_part, target_hash, characters):
    # Try all combinations of 0, 1, 2, or 3 characters (since max password length is 9)
    for length in range(0, 4):
        # 0 to 3 remaining characters
        for combo in itertools.product(characters, repeat=length):
            # Build the complete password
            password = known_part + "".join(combo)
            # Compute its SHA-1 hash
            hashed_password = hashlib.sha1(password.encode()).hexdigest()
            # Check if it matches the target hash
            if hashed_password == target_hash:
                return password
    return None

Find and print the password
password = find_password(known_part, target_hash, characters)
if password:
    print(f"The password is: {password}")
else:
    print("Password not found.")

```

Explanation: Known Part: We know that the password starts with "Om@nYg".

2.3.2.3.4