Hash Identifier

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

import re
import sys
import hashlib
from collections import defaultdict

def identify_hash(hash_string):
    """Identify the type of hash based on pattern, length, and character set."""

    # Clean the hash string
    hash_string = hash_string.strip()

    # Check for empty string
    if not hash_string:
        return "Empty string"

    # Check for common hash formats with special syntax
    if hash_string.startswith('$1$'):
        return "MD5 (Unix)"
    if hash_string.startswith('$2a$') or hash_string.startswith('$2b$') or
hash_string.startswith('$2y$'):
        return "Bcrypt"
    if hash_string.startswith('$5$'):
        return "SHA-256 (Unix)"
    if hash_string.startswith('$6$'):
        return "SHA-512 (Unix)"
    if hash_string.startswith('$pbkdf2-sha256$'):
        return "PBKDF2-SHA256"
    if hash_string.startswith('$sha1$'):
        return "SHA-1 (Unix)"
    if hash_string.startswith('$pdf$'):
        return "PDF (Hashcat format)"
    if hash_string.startswith('$P$') or hash_string.startswith('$H$'):
        return "PHPass (WordPress/phpBB)"
    if hash_string.startswith('$apr1$'):
```

```
    return "APR1-MD5"
if re.match(r'^[a-fA-F0-9]{32}:[a-fA-F0-9]{32}$', hash_string):
    return "MD5(Half:Salt)"

# Check for common hash lengths
hash_length = len(hash_string)
possible_types = []

# Check if the hash is hexadecimal
if re.match(r'^[a-fA-F0-9]+$', hash_string):
    if hash_length == 32:
        possible_types.append("MD5")
        possible_types.append("MD4")
        possible_types.append("NTLM")
        possible_types.append("RIPEMD-128")
    elif hash_length == 40:
        possible_types.append("SHA-1")
        possible_types.append("RIPEMD-160")
    elif hash_length == 64:
        possible_types.append("SHA-256")
        possible_types.append("RIPEMD-256")
    elif hash_length == 96:
        possible_types.append("SHA-384")
    elif hash_length == 128:
        possible_types.append("SHA-512")
        possible_types.append("Whirlpool")
    elif hash_length == 16:
        possible_types.append("MySQL323")
        possible_types.append("DES(Oracle)")
    elif hash_length == 41 and hash_string.startswith('*'):
        possible_types.append("MySQL5")
    elif hash_length == 56:
        possible_types.append("SHA-224")
    elif hash_length == 8:
        possible_types.append("CRC32")
        possible_types.append("ADLER32")

# Check for Base64 character set (with potential padding)
if re.match(r'^[A-Za-z0-9+/]+={0,2}$', hash_string):
    if hash_length == 24:
        possible_types.append("MD5 (Base64)")
```

```
        elif hash_length == 28:
            possible_types.append("SHA-1 (Base64)")
        elif hash_length == 44:
            possible_types.append("SHA-256 (Base64)")
        elif hash_length == 88:
            possible_types.append("SHA-512 (Base64)")
        else:
            possible_types.append("Base64 encoded")

    # No specific hash type identified, give general suggestion
    if not possible_types:
        if re.match(r'^[a-fA-F0-9]+$', hash_string):
            return f"Unknown hash (Hexadecimal, {hash_length} chars)"
        else:
            return f"Unknown format (possibly not a standard hash, or custom format)"

    return " or ".join(possible_types)

def main():
    if len(sys.argv) != 2:
        print("Usage: python hash_identifier.py <hash_file>")
        sys.exit(1)

    hash_file = sys.argv[1]

    try:
        with open(hash_file, 'r') as f:
            lines = f.readlines()

        print(f"Analyzing {len(lines)} hashes from {hash_file}...\n")

        hash_types = defaultdict(int)

        for i, line in enumerate(lines, 1):
            hash_string = line.strip()
            if not hash_string or hash_string.startswith('#'):
                continue

            hash_type = identify_hash(hash_string)
            hash_types[hash_type] += 1
```

```
            # Print the first few and last few hash identifications
            if i <= 3 or i > len(lines) - 3:
                print(f"Line {i}: {hash_string[:40]}{'...' if len(hash_string) > 40 else ''} -
> {hash_type}")
            elif i == 4 and len(lines) > 6:
                print(f"... ({len(lines) - 6} more hashes) ...")

        print("\nSummary of hash types:")
        for hash_type, count in sorted(hash_types.items(), key=lambda x: x[1], reverse=True):
            print(f"  {hash_type}: {count}")

    except FileNotFoundError:
        print(f"Error: File '{hash_file}' not found.")
        sys.exit(1)
    except Exception as e:
        print(f"Error: {e}")
        sys.exit(1)

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

Revision #1

Created 2025-11-25 18:04:30 UTC by David Rizzo

Updated 2025-11-25 18:04:43 UTC by David Rizzo