HashTag.py

The Parsing and Identification of Password Hashes

Introduction

HashTag.py is a tool written in python which parses and identifies various password hashes based on their type. HashTag was inspired by attending PasswordsCon 13 in Las Vegas, KoreLogic's 'Crack Me If You Can' competition at Defcon, and the research of iphelix and his toolkit PACK. HashTag supports the identification of over 250 hash types along with matching them to over 110 hashcat modes. HashTag is able to identify a single hash, parse a single file and identify the hashes within it, or traverse a root directory and all subdirectories for potential hash files and identify any hashes found.

Hash Identification

One of the biggest aspects of this tool is the identification of password hashes. The main attributes I used to distinguish between hash types are character set (hexadecimal, alphanumeric, etc.), hash length, hash format (e.g. 32 character hash followed by a colon and a salt), and any specific substrings (e.g. '1'). A lot of password hash strings can't be identified as one specific hash type based on these attributes. For example, MD5 and NTLM hashes are both 32 character hexadecimal strings. In these cases I make an exhaustive list of possible types and have the tool output reflect that. During development I created an excel spreadsheet which contains much of the hash information I used:


Hash List.xlsx

Usage

HashTag.py {-sh hash |-f file |-d directory} [-o output_filename] [-hc] [-n]

-h, --help	show this help message and exit
-sh SINGLEHASH, --singleHash SINGLEHASH	Identify a single hash
-f FILE, --file FILE	Parse a single file for hashes and identify them
-d DIRECTORY, --directory DIRECTORY	Parse, identify, and categorize hashes within a directory and all subdirectories
-o OUTPUT, --output OUTPUT	Filename to output full list of all identified hashes. --file default filename: HashTag/HashTag_Output_File.txt --directory default filename: HashTag/HashTag_Hash_File.txt
-hc, --hashcatOutput	--file: Output a file per different hash type found, if corresponding hashcat mode exists --directory: Appends hashcat mode to end of separate files
-n, --notFound	--file: Include unidentifiable hashes in the output file. Good for tool debugging (Is it Identifying properly?)

Identify a Hash

HashTag.py -sh 1MtCReiOj$zvOdxVzPtrQ.PXNW3hTHI0

```
C:\Python27>python HashTag.py -sh $1$MtCReiOj$zvOdxVzPtrQ.PXNW3hTHI0

Hash: $1$MtCReiOj$zvOdxVzPtrQ.PXNW3hTHI0

[*] md5crypt, MD5(Unix), FreeBSD MD5, Cisco-IOS MD5 - Hashcat Mode 500
```

HashTag.py -sh 7026360f1826f8bc

```
C:\Python27>python HashTag.py -sh 7026360f1826f8bc

Hash: 7026360f1826f8bc

[*] MySQL, MySQL323
[*] Oracle 7-10g, DES(Oracle) - Hashcat Mode 3100
[*] CRC-64
[*] SAPB
[*] substr(md5($pass),0,16)
[*] substr(md5($pass),16,16)
[*] substr(md5($pass),8,16)
```

HashTag.py -sh 3b1015ccf38fc2a32c18674c166fa447

```
C:\Python27>python HashTag.py -sh 3b1015ccf38fc2a32c18674c166fa447

Hash: 3b1015ccf38fc2a32c18674c166fa447

[*] MD5 - Hashcat Mode 0
[*] NTLM - Hashcat Mode 1000
[*] MD4 - Hashcat Mode 900
[*] LM - Hashcat Mode 3000
[*] RAdmin v2.x
[*] Haval-128
[*] MD2
[*] RipeMD-128
[*] Tiger-128
[*] Snefru-128
[*] MD5(HMAC)
[*] MD4(HMAC)
[*] Haval-128(HMAC)
[*] RipeMD-128(HMAC)
[*] Tiger-128(HMAC)
[*] Snefru-128(HMAC)
[*] MD2(HMAC)
[*] MD5(ZipMonster)
[*] MD5(HMAC(Wordpress))
[*] Skein-256(128)
[*] Skein-512(128)
[*] md5($pass.$salt) - Hashcat Mode 10
[*] md5($pass.$salt.$pass)
[*] md5($pass.md5($pass))
[*] md5($salt.$pass) - Hashcat Mode 20
[*] md5($salt.$pass.$salt) - Hashcat Mode 3810
[*] md5($salt.$pass.$username)
[*] md5($salt.'-'.md5($pass))
[*] md5($salt.md5($pass)) - Hashcat Mode 3710
[*] md5($salt.md5($pass).$salt)
[*] md5($salt.MD5($pass).$username)
[*] md5($salt.md5($pass.$salt)) - Hashcat Mode 4110
[*] md5($salt.md5($salt.$pass)) - Hashcat Mode 4010
[*] md5($salt.md5(md5($pass).$salt))
[*] md5($username.0.$pass) - Hashcat Mode 4210
[*] md5($username.LF.$pass)
[*] md5($username.md5($pass).$salt)
```

Parsing and Identifying Hashes from a File

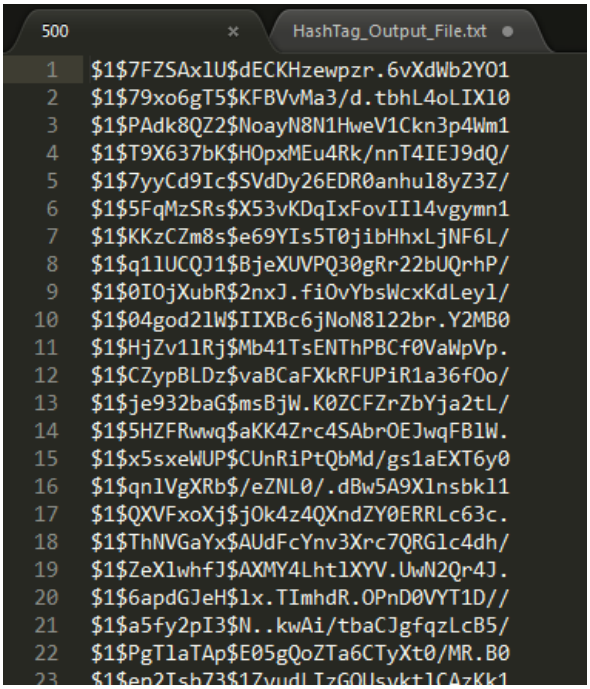
HashTag.py -f testdir\street-hashes.10.txt -hc

```
C:\Python27>python HashTag.py -f testdir\street-hashes.10.txt -hc

File Mimetype: text/plain
Hashes Found: 866
File successfully written: HashTag\HashTag_Output_File.txt
```

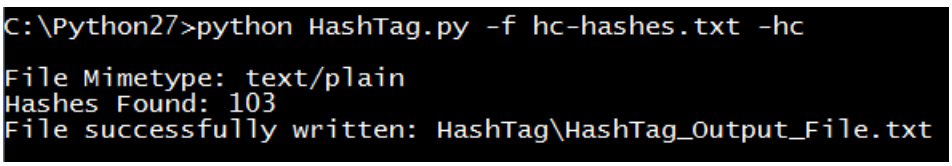
Output:

Each identified hash outputs the hash, char length, hashcat modes (if found) , and possible hash types.

Using the -hc/--hashcat argument we get a file for each hash type if a corresponding hashcat mode is found. This makes the process of cracking hashes with hashcat much easier as you immediately have the mode and input file of hashes.



HashTag.py -f hc-hashes -hc

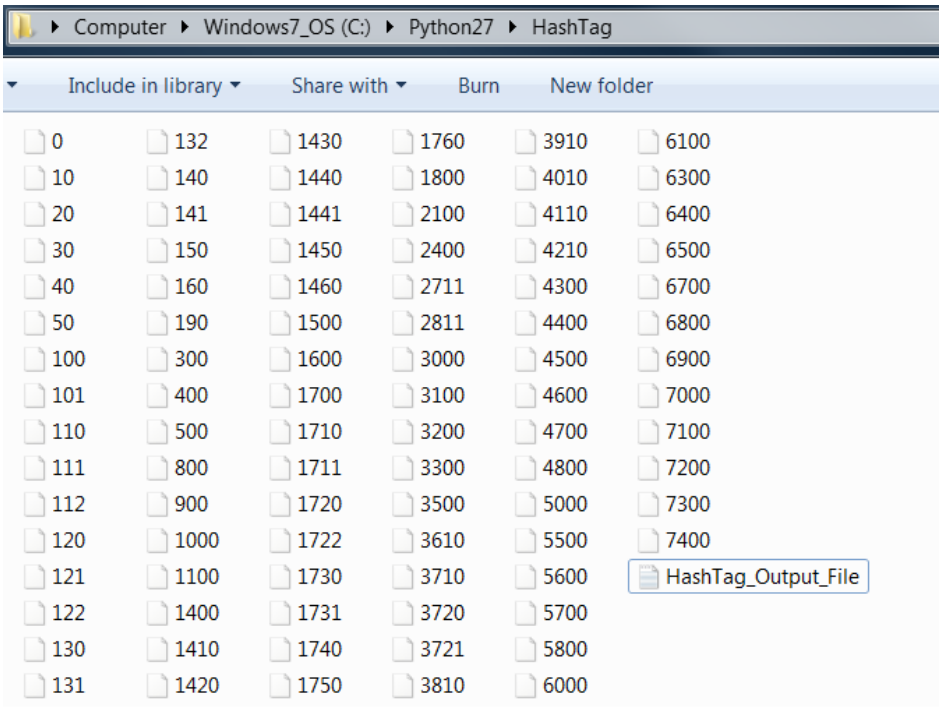


Output:

HashTag_Output_File.txt

```
Hash: 0x0100saltfc9def6a4f847576458241bf11c2a906140b1db2fc9def6a4f847576458241bf11c2a906140b1db2
Char Length: 90
Hashcat Modes: ['131']
Hash Types: ['MSSQL(2000)']

Hash: u4-netntlm::kNS:338d08f8e26de933000000000000000000000000000000:9526fb8c23a90751cdd619b6cea564742e1e4bf33
Char Length: 130
Hashcat Modes: ['5500', '5600']
Hash Types: ['NetNTLMv2', 'NetNTLMv1-VANILLA / NetNTLMv1+ESS']

Hash: 265f92cc8dac35bf09694ff921228ef8:500:lp@trash-mail.com
Char Length: 54
Hashcat Modes: ['6800']
Hash Types: ['Lastpass']

Hash: {ssha256}06$aJckFGJAB30LTe10$ohUsB7LBPlgclE3hJg9x042DLJvQyxVCX.nZZLEz.g2
Char Length: 72
Hashcat Modes: ['6400']
Hash Types: ['AIX {ssha256}']

Hash: AK1AAECAwQFBgcICRARNGqgeC3is8gv2xWWRony9NJnDgEA
Char Length: 47
Hashcat Modes: ['7000']
Hash Types: ['Fortigate (FortiOS)']

Hash: $episerver$*0*bEtiVGhPNlZpcUN4a3ExTg==*utkfN0EOgljbv5FoZ6+AcZD5iLk
Char Length: 66
Hashcat Modes: ['141']
Hash Types: ['EPiServer 6.x < v4']

Hash: 250920b3a5e31318806a032a4674df7e:1234
Char Length: 37
Hashcat Modes: ['30', '40', '50', '2100', '2711', '2811', '3610', '3710', '3720', '3721', '3810', '3910', '4010'
Hash Types: ['md5($salt.$pass.$salt)', 'md5($salt.md5($pass))', 'md5($salt.md5($pass.$salt))', 'md5($salt.md5($s
).md5($salt))', 'md5(md5($salt).$pass)', 'HMAC-MD5 (key = $pass)', 'HMAC-MD5 (key = $salt)', 'md5($pass.md5($sal
unicode($pass).$salt)', 'Domain Cached Credentials2, mscash2', 'md5($salt.unicode($pass))', 'vBulletin > v3.8.5'
```

Again, using the -hc/--hashcat argument we get a file for each hash type if a corresponding hashcat mode is found. This is useful if you have a large file with different types of hashes.



Traversing Directories and Identifying Hashes

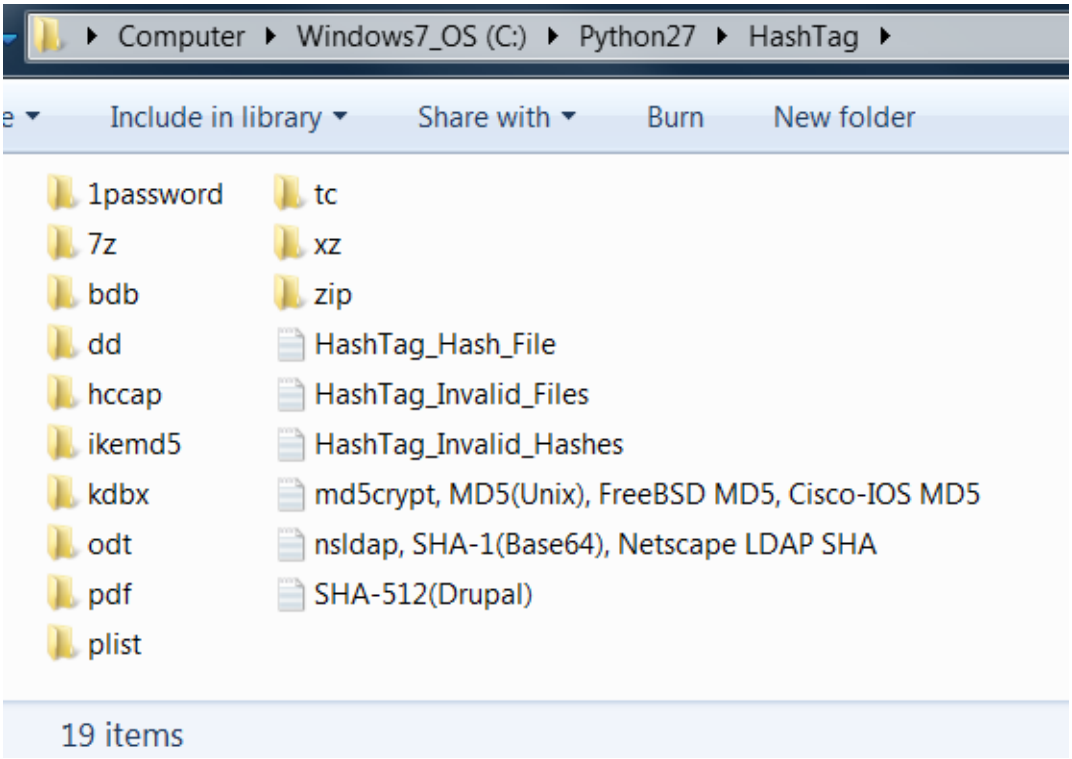
HashTag.py -d ./testdir -hc

```
C:\Python27>python HashTag.py -d ./testdir

Total Hashes Found: 8104
Valid file types: 3
Invalid file types: 3

Now identifying 8104 hashes from 3 files...
810/8104 hashes have been identified and written.
1620/8104 hashes have been identified and written.
2430/8104 hashes have been identified and written.
3240/8104 hashes have been identified and written.
4050/8104 hashes have been identified and written.
4860/8104 hashes have been identified and written.
5670/8104 hashes have been identified and written.
6480/8104 hashes have been identified and written.
7290/8104 hashes have been identified and written.
8100/8104 hashes have been identified and written.

8104 hashes have been identified and written to separate files based on hash type.
A full list was written to file HashTag\HashTag_Hash_File.txt
```

Output:

There are three main things included in the output:

- Folders containing copies of potentially password protected files. This makes it easy to group files based on extension and attempt to crack them.
- HashTag default files - List of all hashes, password protected files the tool doesn't recognize, and hashes the tool can't identify (good for tool debugging).
- Files for each identified hash type - each file contains a list of hashes. The -hc/--hashcat argument will append the hashcat mode (if found) to the filename.



Resources

Quite a bit of research went into different password hash types. During this research I found a script called Hash Identifier which was actually included in one of the Backtrack versions. After looking it over I feel my tool has a lot more functionality, efficiency, and accuracy. My other research ranged from finding different hash examples to generating my own via passlib. I would like to give credit to the following resources which all had some impact in the creation of this tool.

http://wiki.insidepro.com/index.php/Main_Page

https://hashcat.net/wiki/

http://openwall.info/wiki/john/

http://pythonhosted.org/passlib/index.html